

- ▶ **Agenda:** explore the **Advanced Encryption Standard (AES)**, i.e.,

Square [4] \rightsquigarrow **Rijndael** [5] \rightsquigarrow **AES** [2, 8],

via

1. an “in theory”, i.e., design-oriented perspective, and
2. an “in practice”, i.e., implementation-oriented perspective,

- ▶ **Caveat!**

~ 2 hours \Rightarrow introductory, and (very) selective (versus definitive) coverage.

Part 1: in theory (1)

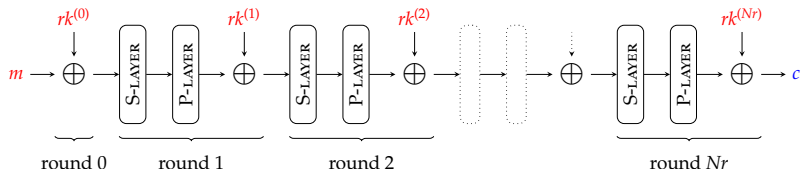
Specification

- ▶ AES is an (iterated) block cipher, where

$$\text{ENC} : \{0, 1\}^{8 \cdot 4 \cdot Nk} \times \{0, 1\}^{8 \cdot 4 \cdot Nb} \rightarrow \{0, 1\}^{8 \cdot 4 \cdot Nb}$$

$$\text{DEC} : \{0, 1\}^{8 \cdot 4 \cdot Nb} \times \{0, 1\}^{8 \cdot 4 \cdot Nb} \rightarrow \{0, 1\}^{8 \cdot 4 \cdot Nk}$$

are realised by using a **substitution-permutation network**



i.e., $Nr + 1$ **rounds**: each r -th round

- ▶ applies one or more **round functions**,
- ▶ involves a **round key** $rk^{(r)}$ derived from the **cipher key** k .

Part 1: in theory (2)

Specification

- ▶ AES is actually a *family* of block ciphers: per [8, Figure 3], the parameter sets are

	Nk	Nb	Nr
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

but we'll focus *exclusively* on AES-128 encryption only.

Part 1: in theory (3)

Low(er)-level concepts/components

- ▶ AES [8, Section 4] operates on elements in the **finite field** \mathbb{F}_{2^8} , which is realised concretely as

$$\mathbb{F}_2[\mathbf{x}]/p(\mathbf{x})$$

where

$$p(\mathbf{x}) = \mathbf{x}^8 + \mathbf{x}^4 + \mathbf{x}^3 + \mathbf{x} + 1$$

meaning

- ▶ a given field element is represented using a polynomial whose **indeterminate** is \mathbf{x} ,
- ▶ coefficients of such polynomials are in the field \mathbb{F}_2 , and
- ▶ arithmetic with field elements is modulo an **irreducible polynomial** $p(\mathbf{x}) = \mathbf{x}^8 + \mathbf{x}^4 + \mathbf{x}^3 + \mathbf{x} + 1$.

Part 1: in theory (4)

Low(er)-level concepts/components

► Beware!

- a hexadecimal short-hand is used for immediate field elements, e.g.,

$$13 \mapsto 13_{(16)} \equiv 00010011_{(2)} \equiv \langle 1, 1, 0, 0, 1, 0, 0, 0 \rangle_{(x)} \mapsto x^4 + x + 1,$$

and

- to avoid confusion, we carefully highlight where arithmetic in some field F is required, e.g.,

\oplus_F	\mapsto	“addition in the field F ”
\ominus_F	\mapsto	“subtraction in the field F ”
\otimes_F	\mapsto	“multiplication in the field F ”
\oslash_F	\mapsto	“division in the field F ”

► Beware!

► we can ignore/avoid *most* field arithmetic, but rely on at least:

1. **addition**, i.e.,

$$r = x \oplus_{\mathbb{F}_{2^8}} y \mapsto x \oplus y.$$

2. **multiplication-by-x**, i.e.,

$$r = \text{xtimes}(x) = \mathbf{x} \otimes_{\mathbb{F}_{2^8}} x \mapsto \begin{cases} \langle 0, x_0, x_1, x_2, x_3, x_4, x_5, x_6 \rangle \oplus \langle 1, 1, 0, 1, 1, 0, 0, 0 \rangle & \text{if } x_7 = 1 \\ \langle 0, x_0, x_1, x_2, x_3, x_4, x_5, x_6 \rangle & \text{otherwise} \end{cases}$$

3. **multiplication-by-c**, e.g.,

$$\begin{aligned} r &= \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x &= & 1 \otimes_{\mathbb{F}_{2^8}} x &= & x \\ r &= \mathbf{02} \otimes_{\mathbb{F}_{2^8}} x &= & \mathbf{x} \otimes_{\mathbb{F}_{2^8}} x &= & \text{xtimes}(x) \\ r &= \mathbf{03} \otimes_{\mathbb{F}_{2^8}} x &= & (\mathbf{x} + 1) \otimes_{\mathbb{F}_{2^8}} x &= & \text{xtimes}(x) \oplus x \end{aligned}$$

► Beware!

► we can ignore/avoid *most* field arithmetic, but rely on at least:

1. **addition**, i.e.,

$$r = x \oplus_{\mathbb{F}_{2^8}} y \mapsto x \oplus y.$$

2. **multiplication-by-x**, i.e.,

$$r = \text{xtimes}(x) = x \otimes_{\mathbb{F}_{2^8}} x \mapsto \begin{cases} (x \ll 1) \oplus 11\text{B} & \text{if } x_7 = 1 \\ x \ll 1 & \text{otherwise} \end{cases}$$

3. **multiplication-by-c**, e.g.,

$$\begin{aligned} r &= \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x &= & 1 \otimes_{\mathbb{F}_{2^8}} x &= & x \\ r &= \mathbf{02} \otimes_{\mathbb{F}_{2^8}} x &= & \mathbf{x} \otimes_{\mathbb{F}_{2^8}} x &= & \text{xtimes}(x) \\ r &= \mathbf{03} \otimes_{\mathbb{F}_{2^8}} x &= & (\mathbf{x} + 1) \otimes_{\mathbb{F}_{2^8}} x &= & \text{xtimes}(x) \oplus x \end{aligned}$$

Part 1: in theory (5)

Low(er)-level concepts/components

► AES [8, Section 3.4] organises field elements into (4×4) -element matrices, e.g.,

1. the r -th state matrix

$$s^{(r)} = \begin{bmatrix} s_{0,0}^{(r)} & s_{0,1}^{(r)} & s_{0,2}^{(r)} & s_{0,3}^{(r)} \\ s_{1,0}^{(r)} & s_{1,1}^{(r)} & s_{1,2}^{(r)} & s_{1,3}^{(r)} \\ s_{2,0}^{(r)} & s_{2,1}^{(r)} & s_{2,2}^{(r)} & s_{2,3}^{(r)} \\ s_{3,0}^{(r)} & s_{3,1}^{(r)} & s_{3,2}^{(r)} & s_{3,3}^{(r)} \end{bmatrix}$$

for each $s_{ij}^{(r)} \in \mathbb{F}_{2^8}$, or

2. the r -th round key matrix

$$rk^{(r)} = \begin{bmatrix} rk_{0,0}^{(r)} & rk_{0,1}^{(r)} & rk_{0,2}^{(r)} & rk_{0,3}^{(r)} \\ rk_{1,0}^{(r)} & rk_{1,1}^{(r)} & rk_{1,2}^{(r)} & rk_{1,3}^{(r)} \\ rk_{2,0}^{(r)} & rk_{2,1}^{(r)} & rk_{2,2}^{(r)} & rk_{2,3}^{(r)} \\ rk_{3,0}^{(r)} & rk_{3,1}^{(r)} & rk_{3,2}^{(r)} & rk_{3,3}^{(r)} \end{bmatrix}$$

for each $rk_{ij}^{(r)} \in \mathbb{F}_{2^8}$,

which can be read from (resp. written to) in column-wise order.

- ▶ AES [8, Section 5.1.1] uses a single S-box, defined as the composition of two functions, i.e.,

$$\text{S-BOX}(x) = (f \circ g)(x) = f(g(x)),$$

where

$$g(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 \otimes_{\mathbb{F}_2^8} x & \text{otherwise} \end{cases}$$

i.e., g is a field (pseudo-)inversion, and

$$f \left(\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \right) = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \otimes_{\mathbb{F}_2} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \oplus_{\mathbb{F}_2} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

is an affine transformation.

Part 1: in theory (6)

Low(er)-level concepts/components

- ▶ AES [8, Section 5.1.1] uses a single S-box, defined as the composition of two functions, i.e.,

$$\text{S-BOX}(x) = (f \circ g)(x) = f(g(x)),$$

where

$$g(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 \otimes_{\mathbb{F}_2^8} x & \text{otherwise} \end{cases}$$

i.e., g is a field (pseudo-)inversion, and

$$f \left(\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \right) = \begin{bmatrix} x_0 \oplus_{\mathbb{F}_2} & & & & x_4 \oplus_{\mathbb{F}_2} & x_5 \oplus_{\mathbb{F}_2} & x_6 \oplus_{\mathbb{F}_2} & x_7 \oplus_{\mathbb{F}_2} & 1 \\ x_0 \oplus_{\mathbb{F}_2} & x_1 \oplus_{\mathbb{F}_2} & & & & x_5 \oplus_{\mathbb{F}_2} & x_6 \oplus_{\mathbb{F}_2} & x_7 \oplus_{\mathbb{F}_2} & 1 \\ x_0 \oplus_{\mathbb{F}_2} & x_1 \oplus_{\mathbb{F}_2} & x_2 \oplus_{\mathbb{F}_2} & & & & x_6 \oplus_{\mathbb{F}_2} & x_7 \oplus_{\mathbb{F}_2} & 0 \\ x_0 \oplus_{\mathbb{F}_2} & x_1 \oplus_{\mathbb{F}_2} & x_2 \oplus_{\mathbb{F}_2} & x_3 \oplus_{\mathbb{F}_2} & & & & x_7 \oplus_{\mathbb{F}_2} & 0 \\ x_0 \oplus_{\mathbb{F}_2} & x_1 \oplus_{\mathbb{F}_2} & x_2 \oplus_{\mathbb{F}_2} & x_3 \oplus_{\mathbb{F}_2} & x_4 & & & & \oplus_{\mathbb{F}_2} 0 \\ & x_1 \oplus_{\mathbb{F}_2} & x_2 \oplus_{\mathbb{F}_2} & x_3 \oplus_{\mathbb{F}_2} & x_4 \oplus_{\mathbb{F}_2} & x_5 & & & \oplus_{\mathbb{F}_2} 1 \\ & & x_2 \oplus_{\mathbb{F}_2} & x_3 \oplus_{\mathbb{F}_2} & x_4 \oplus_{\mathbb{F}_2} & x_5 \oplus_{\mathbb{F}_2} & x_6 & & \oplus_{\mathbb{F}_2} 1 \\ & & & x_3 \oplus_{\mathbb{F}_2} & x_4 \oplus_{\mathbb{F}_2} & x_5 \oplus_{\mathbb{F}_2} & x_6 \oplus_{\mathbb{F}_2} & x_7 \oplus_{\mathbb{F}_2} & 0 \end{bmatrix}$$

is an affine transformation.

Algorithm (AES round functions [8, Section 5.1])

Input: A state matrix $s^{(r)}$, and a round key matrix $rk^{(r)}$

Output: A state matrix $s'^{(r)} = \text{AddRoundKey}(s^{(r)}, rk^{(r)})$

$$s'^{(r)} = \text{AddRoundKey}(s^{(r)}, rk^{(r)})$$

$$= \text{AddRoundKey} \left(\begin{bmatrix} s_{0,0}^{(r)} & s_{0,1}^{(r)} & s_{0,2}^{(r)} & s_{0,3}^{(r)} \\ s_{1,0}^{(r)} & s_{1,1}^{(r)} & s_{1,2}^{(r)} & s_{1,3}^{(r)} \\ s_{2,0}^{(r)} & s_{2,1}^{(r)} & s_{2,2}^{(r)} & s_{2,3}^{(r)} \\ s_{3,0}^{(r)} & s_{3,1}^{(r)} & s_{3,2}^{(r)} & s_{3,3}^{(r)} \end{bmatrix}, \begin{bmatrix} rk_{0,0}^{(r)} & rk_{0,1}^{(r)} & rk_{0,2}^{(r)} & rk_{0,3}^{(r)} \\ rk_{1,0}^{(r)} & rk_{1,1}^{(r)} & rk_{1,2}^{(r)} & rk_{1,3}^{(r)} \\ rk_{2,0}^{(r)} & rk_{2,1}^{(r)} & rk_{2,2}^{(r)} & rk_{2,3}^{(r)} \\ rk_{3,0}^{(r)} & rk_{3,1}^{(r)} & rk_{3,2}^{(r)} & rk_{3,3}^{(r)} \end{bmatrix} \right)$$

$$= \begin{bmatrix} \left(s_{0,0}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{0,0}^{(r)} \right) & \left(s_{0,1}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{0,1}^{(r)} \right) & \left(s_{0,2}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{0,2}^{(r)} \right) & \left(s_{0,3}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{0,3}^{(r)} \right) \\ \left(s_{1,0}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{1,0}^{(r)} \right) & \left(s_{1,1}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{1,1}^{(r)} \right) & \left(s_{1,2}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{1,2}^{(r)} \right) & \left(s_{1,3}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{1,3}^{(r)} \right) \\ \left(s_{2,0}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{2,0}^{(r)} \right) & \left(s_{2,1}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{2,1}^{(r)} \right) & \left(s_{2,2}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{2,2}^{(r)} \right) & \left(s_{2,3}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{2,3}^{(r)} \right) \\ \left(s_{3,0}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{3,0}^{(r)} \right) & \left(s_{3,1}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{3,1}^{(r)} \right) & \left(s_{3,2}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{3,2}^{(r)} \right) & \left(s_{3,3}^{(r)} \oplus_{\mathbb{F}_{2^8}} rk_{3,3}^{(r)} \right) \end{bmatrix}$$

Algorithm (AES round functions [8, Section 5.1])

Input: A state matrix $s^{(r)}$

Output: A state matrix $s'^{(r)} = \text{SubBytes}(s^{(r)})$

$$s'^{(r)} = \text{SubBytes}(s^{(r)})$$

$$= \text{SubBytes} \left(\begin{bmatrix} s_{0,0}^{(r)} & s_{0,1}^{(r)} & s_{0,2}^{(r)} & s_{0,3}^{(r)} \\ s_{1,0}^{(r)} & s_{1,1}^{(r)} & s_{1,2}^{(r)} & s_{1,3}^{(r)} \\ s_{2,0}^{(r)} & s_{2,1}^{(r)} & s_{2,2}^{(r)} & s_{2,3}^{(r)} \\ s_{3,0}^{(r)} & s_{3,1}^{(r)} & s_{3,2}^{(r)} & s_{3,3}^{(r)} \end{bmatrix} \right)$$

$$= \begin{bmatrix} \text{S-BOX} \left(s_{0,0}^{(r)} \right) & \text{S-BOX} \left(s_{0,1}^{(r)} \right) & \text{S-BOX} \left(s_{0,2}^{(r)} \right) & \text{S-BOX} \left(s_{0,3}^{(r)} \right) \\ \text{S-BOX} \left(s_{1,0}^{(r)} \right) & \text{S-BOX} \left(s_{1,1}^{(r)} \right) & \text{S-BOX} \left(s_{1,2}^{(r)} \right) & \text{S-BOX} \left(s_{1,3}^{(r)} \right) \\ \text{S-BOX} \left(s_{2,0}^{(r)} \right) & \text{S-BOX} \left(s_{2,1}^{(r)} \right) & \text{S-BOX} \left(s_{2,2}^{(r)} \right) & \text{S-BOX} \left(s_{2,3}^{(r)} \right) \\ \text{S-BOX} \left(s_{3,0}^{(r)} \right) & \text{S-BOX} \left(s_{3,1}^{(r)} \right) & \text{S-BOX} \left(s_{3,2}^{(r)} \right) & \text{S-BOX} \left(s_{3,3}^{(r)} \right) \end{bmatrix}$$

Algorithm (AES round functions [8, Section 5.1])

Input: A state matrix $s^{(r)}$

Output: A state matrix $s'^{(r)} = \text{ShiftRows}(s^{(r)})$

$$s'^{(r)} = \text{ShiftRows}(s^{(r)})$$

$$= \text{ShiftRows} \left(\begin{bmatrix} s_{0,0}^{(r)} & s_{0,1}^{(r)} & s_{0,2}^{(r)} & s_{0,3}^{(r)} \\ s_{1,0}^{(r)} & s_{1,1}^{(r)} & s_{1,2}^{(r)} & s_{1,3}^{(r)} \\ s_{2,0}^{(r)} & s_{2,1}^{(r)} & s_{2,2}^{(r)} & s_{2,3}^{(r)} \\ s_{3,0}^{(r)} & s_{3,1}^{(r)} & s_{3,2}^{(r)} & s_{3,3}^{(r)} \end{bmatrix} \right)$$

$$= \begin{bmatrix} s_{0,0}^{(r)} & s_{0,1}^{(r)} & s_{0,2}^{(r)} & s_{0,3}^{(r)} \\ s_{1,1}^{(r)} & s_{1,2}^{(r)} & s_{1,3}^{(r)} & s_{1,0}^{(r)} \\ s_{2,2}^{(r)} & s_{2,3}^{(r)} & s_{2,0}^{(r)} & s_{2,1}^{(r)} \\ s_{3,3}^{(r)} & s_{3,0}^{(r)} & s_{3,1}^{(r)} & s_{3,2}^{(r)} \end{bmatrix}$$

Algorithm (AES round functions [8, Section 5.1])

Input: A state matrix $s^{(r)}$

Output: A state matrix $s'^{(r)} = \text{MixColumns}(s^{(r)})$

$$s'^{(r)} = \text{MixColumns}(s^{(r)})$$

$$= \text{MixColumns} \begin{pmatrix} s_{0,0}^{(r)} & s_{0,1}^{(r)} & s_{0,2}^{(r)} & s_{0,3}^{(r)} \\ s_{1,0}^{(r)} & s_{1,1}^{(r)} & s_{1,2}^{(r)} & s_{1,3}^{(r)} \\ s_{2,0}^{(r)} & s_{2,1}^{(r)} & s_{2,2}^{(r)} & s_{2,3}^{(r)} \\ s_{3,0}^{(r)} & s_{3,1}^{(r)} & s_{3,2}^{(r)} & s_{3,3}^{(r)} \end{pmatrix}$$

where each column of the result, i.e., for each $0 \leq j < 4$, is produced via

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = \text{MixColumn} \begin{pmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{pmatrix} = \begin{bmatrix} \mathbf{02} & \mathbf{03} & \mathbf{01} & \mathbf{01} \\ \mathbf{01} & \mathbf{02} & \mathbf{03} & \mathbf{01} \\ \mathbf{01} & \mathbf{01} & \mathbf{02} & \mathbf{03} \\ \mathbf{03} & \mathbf{01} & \mathbf{01} & \mathbf{02} \end{bmatrix} \otimes_{\mathbb{F}_{2^8}} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix}$$

$$= \begin{bmatrix} (\mathbf{02} \otimes_{\mathbb{F}_{2^8}} s_{0,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{03} \otimes_{\mathbb{F}_{2^8}} s_{1,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} s_{2,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} s_{3,j}) \\ (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} s_{0,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{02} \otimes_{\mathbb{F}_{2^8}} s_{1,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{03} \otimes_{\mathbb{F}_{2^8}} s_{2,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} s_{3,j}) \\ (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} s_{0,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} s_{1,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{02} \otimes_{\mathbb{F}_{2^8}} s_{2,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{03} \otimes_{\mathbb{F}_{2^8}} s_{3,j}) \\ (\mathbf{03} \otimes_{\mathbb{F}_{2^8}} s_{0,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} s_{1,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} s_{2,j}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{02} \otimes_{\mathbb{F}_{2^8}} s_{3,j}) \end{bmatrix}$$

Algorithm (AES-128.ENC-KEYEXPAND [8, Section 5.2])

Input: A 128-bit cipher key k

Output: An 11-element sequence $rk = \text{AES-128.ENC-KEYEXPAND}(k) \simeq \text{ExpandRoundKey}(k)$ of round keys

Generate a sequence of column vectors

$$\begin{array}{c}
 \text{round key } rk^{(0)} \qquad \qquad \qquad \text{round key } rk^{(1)} \\
 \left[\begin{array}{c|c|c|c} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,3} \\ w_{1,0} & w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,0} & w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,0} & w_{3,1} & w_{3,2} & w_{3,3} \end{array} \right] \left[\begin{array}{c|c|c|c} w_{0,4} & w_{0,5} & w_{0,6} & w_{0,7} \\ w_{1,4} & w_{1,5} & w_{1,6} & w_{1,7} \\ w_{2,4} & w_{2,5} & w_{2,6} & w_{2,7} \\ w_{3,4} & w_{3,5} & w_{3,6} & w_{3,7} \end{array} \right] \dots
 \end{array}$$

using the following rules

1. in the j -th column-vector for $0 \leq j < Nk$, we set $w_{i,j} = k_{i,j}$ to match the cipher key,
2. in the j -th column-vector for $Nk \leq j < Nb \cdot (Nr + 1)$ we set

$$\left[\begin{array}{c} w_{0,j} \\ w_{1,j} \\ w_{2,j} \\ w_{3,j} \end{array} \right] = \begin{cases} \left[\begin{array}{c} rc_{j/Nk} \oplus_{\mathbb{F}_{2^8}} \text{S-BOX}(w_{1,j-1}) \oplus_{\mathbb{F}_{2^8}} w_{0,j-Nk} \\ \text{S-BOX}(w_{2,j-1}) \oplus_{\mathbb{F}_{2^8}} w_{1,j-Nk} \\ \text{S-BOX}(w_{3,j-1}) \oplus_{\mathbb{F}_{2^8}} w_{2,j-Nk} \\ \text{S-BOX}(w_{0,j-1}) \oplus_{\mathbb{F}_{2^8}} w_{3,j-Nk} \end{array} \right] & \text{if } j = 0 \pmod{Nk} \\ \left[\begin{array}{c} w_{0,j-1} \oplus_{\mathbb{F}_{2^8}} w_{0,j-Nk} \\ w_{1,j-1} \oplus_{\mathbb{F}_{2^8}} w_{1,j-Nk} \\ w_{2,j-1} \oplus_{\mathbb{F}_{2^8}} w_{2,j-Nk} \\ w_{3,j-1} \oplus_{\mathbb{F}_{2^8}} w_{3,j-Nk} \end{array} \right] & \text{otherwise} \end{cases}$$

then combining them to yield a sequence of round keys, where $rc^{(r)} = x^{r-1}$ denotes the r -th round constant.

Algorithm (AES-128.ENC-KEYEVOLVE [8, Section 5.2])

Input: The r -th round key matrix $rk^{(r)}$ and round constant $rc^{(r)}$

Output: The $(r + 1)$ -th round key matrix $rk^{(r+1)} = \text{AES-128.ENC-KEYEVOLVE}(k) \simeq \text{“EvolveRoundKey”}(rk^{(r)}, rc^{(r)})$

1. Compute

$$\begin{aligned}
 rk_{0,0}^{(r+1)} &\leftarrow rc^{(r)} \oplus_{\mathbb{F}_{2^8}} \text{S-BOX} \left(rk_{1,3}^{(r)} \right) \oplus_{\mathbb{F}_{2^8}} rk_{0,0}^{(r)} \\
 rk_{1,0}^{(r+1)} &\leftarrow \text{S-BOX} \left(rk_{2,3}^{(r)} \right) \oplus_{\mathbb{F}_{2^8}} rk_{1,0}^{(r)} \\
 rk_{2,0}^{(r+1)} &\leftarrow \text{S-BOX} \left(rk_{3,3}^{(r)} \right) \oplus_{\mathbb{F}_{2^8}} rk_{2,0}^{(r)} \\
 rk_{3,0}^{(r+1)} &\leftarrow \text{S-BOX} \left(rk_{0,3}^{(r)} \right) \oplus_{\mathbb{F}_{2^8}} rk_{3,0}^{(r)}
 \end{aligned}$$

2. Compute

$$\begin{aligned}
 rk_{0,1}^{(r+1)} &\leftarrow rk_{0,0}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{0,1}^{(r)} & rk_{1,1}^{(r+1)} &\leftarrow rk_{1,0}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{1,1}^{(r)} \\
 rk_{2,1}^{(r+1)} &\leftarrow rk_{2,0}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{2,1}^{(r)} & rk_{3,1}^{(r+1)} &\leftarrow rk_{3,0}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{3,1}^{(r)} \\
 \\
 rk_{0,2}^{(r+1)} &\leftarrow rk_{0,1}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{0,2}^{(r)} & rk_{1,2}^{(r+1)} &\leftarrow rk_{1,1}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{1,2}^{(r)} \\
 rk_{2,2}^{(r+1)} &\leftarrow rk_{2,1}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{2,2}^{(r)} & rk_{3,2}^{(r+1)} &\leftarrow rk_{3,1}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{3,2}^{(r)} \\
 \\
 rk_{0,3}^{(r+1)} &\leftarrow rk_{0,2}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{0,3}^{(r)} & rk_{1,3}^{(r+1)} &\leftarrow rk_{1,2}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{1,3}^{(r)} \\
 rk_{2,3}^{(r+1)} &\leftarrow rk_{2,2}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{2,3}^{(r)} & rk_{3,3}^{(r+1)} &\leftarrow rk_{3,2}^{(r+1)} \oplus_{\mathbb{F}_{2^8}} rk_{3,3}^{(r)}
 \end{aligned}$$

3. Return $rk^{(r+1)}$.

Part 1: in theory (10)

High(er)-level concepts/components

Algorithm (AES-128.ENC [8, Section 5.1])

Input: A 128-bit cipher key k , and a 128-bit plaintext message m

Output: A 128-bit ciphertext message $c = \text{AES-128.ENC}(k, m)$

```
1  $rk \leftarrow \text{ExpandRoundKey}(k)$ 
2  $s \leftarrow m$ 
3  $s \leftarrow \text{AddRoundKey}(s, rk^{(0)})$ 
4 for  $r = 1$  upto  $Nr - 1$  step  $+1$  do
5    $s \leftarrow \text{SubBytes}(s)$ 
6    $s \leftarrow \text{ShiftRows}(s)$ 
7    $s \leftarrow \text{MixColumns}(s)$ 
8    $s \leftarrow \text{AddRoundKey}(s, rk^{(r)})$ 
9 end
10  $s \leftarrow \text{SubBytes}(s)$ 
11  $s \leftarrow \text{ShiftRows}(s)$ 
12  $s \leftarrow \text{AddRoundKey}(s, rk^{(10)})$ 
13  $c \leftarrow s$ 
14 return  $c$ 
```

Part 1: in theory (10)

High(er)-level concepts/components

Algorithm (AES-128.ENC [8, Section 5.1])

Input: A 128-bit cipher key k , and a 128-bit plaintext message m

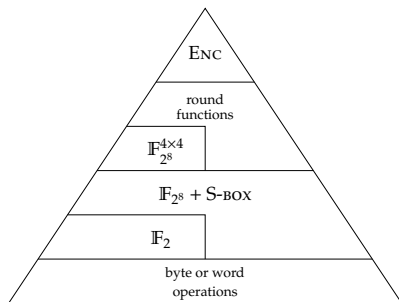
Output: A 128-bit ciphertext message $c = \text{AES-128.ENC}(k, m)$

```
1  $s \leftarrow m$ 
2  $s \leftarrow \text{AddRoundKey}(s, k = rk^{(0)})$ 
3 for  $r = 1$  upto  $Nr - 1$  step  $+1$  do
4    $k \leftarrow \text{EvolveRoundKey}(k, rc^{(r)})$ 
5    $s \leftarrow \text{SubBytes}(s)$ 
6    $s \leftarrow \text{ShiftRows}(s)$ 
7    $s \leftarrow \text{MixColumns}(s)$ 
8    $s \leftarrow \text{AddRoundKey}(s, k = rk^{(r)})$ 
9 end
10  $k \leftarrow \text{EvolveRoundKey}(k, rc^{(10)})$ 
11  $s \leftarrow \text{SubBytes}(s)$ 
12  $s \leftarrow \text{ShiftRows}(s)$ 
13  $s \leftarrow \text{AddRoundKey}(s, k = rk^{(10)})$ 
14  $c \leftarrow s$ 
15 return  $c$ 
```

Part 2: in practice (1)

► Challenge:

- given a functionality “stack”, i.e.,



bridge gap between what we have (bottom) and want (top),

- an **implementation strategy** for doing so must consider many

- goals : parameter set, functionality, ...
- metrics : latency, throughput, memory footprint, ...
- constraints : hardware versus software, data-path width, ...
-
-

Part 2: in practice (7)

Strategy #1

- ▶ **Strategy #1** [2, Section 4.1]:
 - ▶ use pure software (i.e., via ISA),
 - ▶ adopt an unpacked representation of state and round key matrices, using (an array of 16) 8-bit bytes, i.e., instances of type `uint8_t`,
 - ▶ favour use of iterative (i.e., rolled) loops,
 - ▶ compute (or evolve) round keys,
 - ▶ compute round functions.

Part 2: in practice (8)

Strategy #1

- ▶ Selective pre-computation can be effective

multiplication-by-x	: $\mathbb{F}_{2^8} \rightarrow \mathbb{F}_{2^8}$	\leadsto	256B look-up table
division-by-x	: $\mathbb{F}_{2^8} \rightarrow \mathbb{F}_{2^8}$	\leadsto	256B look-up table
round constants	: $\mathbb{Z} \rightarrow \mathbb{F}_{2^8}$	\leadsto	Nr B look-up table
S-box	: $\mathbb{F}_{2^8} \rightarrow \mathbb{F}_{2^8}$	\leadsto	256B look-up table

noting that various (sub-)options exist for the S-box, e.g.,

Pre-compute	Compute	Footprint	Applicability	
			ENC	DEC
	$f \circ g$	0B	✓	✓
g	f	256B	✓	✓
f	g	256B	✓	✗
$f \circ g$		256B	✓	✗

Part 2: in practice (9)

Strategy #1

Listing

```
1 void aes_enc_rnd_key( uint8_t* s, const uint8_t* rk ) {
2     for( int i = 0; i < 16; i++ ) {
3         s[ i ] = s[ i ] ^ rk[ i ];
4     }
5 }
```

Part 2: in practice (9)

Strategy #1

Listing

```
1 void aes_enc_rnd_sub( uint8_t* s ) {  
2     for( int i = 0; i < 16; i++ ) {  
3         s[ i ] = aes_enc_sbox( s[ i ] );  
4     }  
5 }
```

Part 2: in practice (9)

Strategy #1

Listing

```
1 void aes_enc_rnd_row( uint8_t* s ) {
2   AES_ENC_RND_ROW_STEP( 0x1, 0x5, 0x9, 0xD,
3     0xD, 0x1, 0x5, 0x9 );
4   AES_ENC_RND_ROW_STEP( 0x2, 0x6, 0xA, 0xE,
5     0xA, 0xE, 0x2, 0x6 );
6   AES_ENC_RND_ROW_STEP( 0x3, 0x7, 0xB, 0xF,
7     0x7, 0xB, 0xF, 0x3 );
8 }
```

Listing

```
1 #define AES_ENC_RND_ROW_STEP(a,b,c,d,e,f,g,h) { \
2   uint8_t __a1 = s[ a ]; \
3   uint8_t __b1 = s[ b ]; \
4   uint8_t __c1 = s[ c ]; \
5   uint8_t __d1 = s[ d ]; \
6 \
7   s[ e ] = __a1; \
8   s[ f ] = __b1; \
9   s[ g ] = __c1; \
10  s[ h ] = __d1; \
11 }
```


Part 2: in practice (9)

Strategy #1

Listing

```
1 void aes_enc_rnd_mix( uint8_t* s ) {
2   for( int i = 0; i < 4; i++, s += 4 ) {
3     AES_ENC_RND_MIX_STEP( 0x0, 0x1, 0x2, 0x3 );
4   }
5 }
```

Listing

```
1 #define AES_ENC_RND_MIX_STEP(a,b,c,d) { \
2   uint8_t __a1 = s[ a ], __a2 = aes_gf28_mulx( __a1 ); \
3   uint8_t __b1 = s[ b ], __b2 = aes_gf28_mulx( __b1 ); \
4   uint8_t __c1 = s[ c ], __c2 = aes_gf28_mulx( __c1 ); \
5   uint8_t __d1 = s[ d ], __d2 = aes_gf28_mulx( __d1 ); \
6   \
7   uint8_t __a3 = __a1 ^ __a2; \
8   uint8_t __b3 = __b1 ^ __b2; \
9   uint8_t __c3 = __c1 ^ __c2; \
10  uint8_t __d3 = __d1 ^ __d2; \
11  \
12  s[ a ] = __a2 ^ __b3 ^ __c1 ^ __d1; \
13  s[ b ] = __a1 ^ __b2 ^ __c3 ^ __d1; \
14  s[ c ] = __a1 ^ __b1 ^ __c2 ^ __d3; \
15  s[ d ] = __a3 ^ __b1 ^ __c1 ^ __d2; \
16 }
```

Listing

```
1 void aes_enc( uint8_t* c, const uint8_t* m, const uint8_t* k ) {
2     uint8_t s[ 4 * AES_128_NB ];
3     uint8_t rk[ 4 * AES_128_NB ];
4
5     memcpy( s, m, ( 4 * AES_128_NB ) * sizeof( uint8_t ) );
6     memcpy( rk, k, ( 4 * AES_128_NB ) * sizeof( uint8_t ) );
7
8     //      1 initial round
9     aes_enc_rnd_key( s, rk );
10    // Nr - 1 iterated rounds
11    for( int r = 1; r < AES_128_NR; r++ ) {
12        aes_enc_key_evolve( rk, rk, aes_rcon(          r ) );
13        aes_enc_rnd_sub( s          );
14        aes_enc_rnd_row( s          );
15        aes_enc_rnd_mix( s          );
16        aes_enc_rnd_key( s, rk );
17    }
18    //      1 final round
19    aes_enc_key_evolve( rk, rk, aes_rcon( AES_128_NR ) );
20    aes_enc_rnd_sub( s          );
21    aes_enc_rnd_row( s          );
22    aes_enc_rnd_key( s, rk );
23
24    memcpy( c, s, ( 4 * AES_128_NB ) * sizeof( uint8_t ) );
25 }
```

Part 2: in practice (11)

Strategy #2

- ▶ **Strategy #2** [2, Section 4.2] (aka. T-tables):
 - ▶ use pure software (i.e., via ISA),
 - ▶ adopt a column-packed representation of state and round key matrices, using (a set of 4) 32-bit words, i.e., instances of type `uint32_t`,
 - ▶ favour use of straight-line (i.e., unrolled) loops,
 - ▶ pre-compute (or expand) round keys,
 - ▶ pre-compute round functions.

Part 2: in practice (12)

Strategy #2

- ▶ **Idea:** pre-compute most of the round, i.e.,

`MixColumns` ◦ `ShiftRows` ◦ `SubBytes`.

Part 2: in practice (12)

Strategy #2

- **Idea:** pre-compute most of the round, i.e.,

$\text{MixColumns} \circ \text{ShiftRows} \circ \text{SubBytes}$.

- **Step #1:** we already know applying MixColumns will yield

$$\begin{bmatrix} S'_{0,j} \\ S'_{1,j} \\ S'_{2,j} \\ S'_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \otimes_{\mathbb{F}_2^8} \begin{bmatrix} S_{0,j} \\ S_{1,j} \\ S_{2,j} \\ S_{3,j} \end{bmatrix}$$

- **Idea:** pre-compute most of the round, i.e.,

$\text{MixColumns} \circ \text{ShiftRows} \circ \text{SubBytes}.$

- **Step #1:** we already know applying MixColumns will yield

$$\begin{bmatrix} S'_{0,j} \\ S'_{1,j} \\ S'_{2,j} \\ S'_{3,j} \end{bmatrix} = \begin{bmatrix} (\mathbf{02} \otimes_{\mathbb{F}_{2^8}} S_{0,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{03} \otimes_{\mathbb{F}_{2^8}} S_{1,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{2,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{3,j}^{(r)}) \\ (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{0,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{02} \otimes_{\mathbb{F}_{2^8}} S_{1,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{03} \otimes_{\mathbb{F}_{2^8}} S_{2,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{3,j}^{(r)}) \\ (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{0,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{1,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{02} \otimes_{\mathbb{F}_{2^8}} S_{2,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{03} \otimes_{\mathbb{F}_{2^8}} S_{3,j}^{(r)}) \\ (\mathbf{03} \otimes_{\mathbb{F}_{2^8}} S_{0,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{1,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{2,j}^{(r)}) \oplus_{\mathbb{F}_{2^8}} (\mathbf{02} \otimes_{\mathbb{F}_{2^8}} S_{3,j}^{(r)}) \end{bmatrix}$$

- **Idea:** pre-compute most of the round, i.e.,

$\text{MixColumns} \circ \text{ShiftRows} \circ \text{SubBytes}.$

- **Step #1:** we already know applying MixColumns will yield

$$\begin{bmatrix} S'_{0,j}(r) \\ S'_{1,j}(r) \\ S'_{2,j}(r) \\ S'_{3,j}(r) \end{bmatrix} = \begin{bmatrix} \mathbf{02} \otimes_{\mathbb{F}_{2^8}} S_{0,j}(r) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{0,j}(r) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{0,j}(r) \\ \mathbf{03} \otimes_{\mathbb{F}_{2^8}} S_{0,j}(r) \end{bmatrix} \oplus_{\mathbb{F}_{2^8}} \begin{bmatrix} \mathbf{03} \otimes_{\mathbb{F}_{2^8}} S_{1,j}(r) \\ \mathbf{02} \otimes_{\mathbb{F}_{2^8}} S_{1,j}(r) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{1,j}(r) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{1,j}(r) \end{bmatrix} \oplus_{\mathbb{F}_{2^8}} \begin{bmatrix} \mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{2,j}(r) \\ \mathbf{03} \otimes_{\mathbb{F}_{2^8}} S_{2,j}(r) \\ \mathbf{02} \otimes_{\mathbb{F}_{2^8}} S_{2,j}(r) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{2,j}(r) \end{bmatrix} \oplus_{\mathbb{F}_{2^8}} \begin{bmatrix} \mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{3,j}(r) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} S_{3,j}(r) \\ \mathbf{03} \otimes_{\mathbb{F}_{2^8}} S_{3,j}(r) \\ \mathbf{02} \otimes_{\mathbb{F}_{2^8}} S_{3,j}(r) \end{bmatrix}$$

- **Idea:** pre-compute most of the round, i.e.,

MixColumns \circ ShiftRows \circ SubBytes.

- **Step #2:** an equivalent RHS can be formed from pre-computed look-up tables, i.e.,

$$\begin{bmatrix} S'_{0,j} \\ S'_{1,j} \\ S'_{2,j} \\ S'_{3,j} \end{bmatrix} = T_0[s_{0,j}^{(r)}] \oplus_{\mathbb{F}_{2^8}} T_1[s_{1,j}^{(r)}] \oplus_{\mathbb{F}_{2^8}} T_2[s_{2,j}^{(r)}] \oplus_{\mathbb{F}_{2^8}} T_3[s_{3,j}^{(r)}]$$

where

$$T_0[x] = \begin{bmatrix} \mathbf{02} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{03} \otimes_{\mathbb{F}_{2^8}} x \end{bmatrix} \quad T_1[x] = \begin{bmatrix} \mathbf{03} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{02} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x \end{bmatrix}$$

$$T_2[x] = \begin{bmatrix} \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{03} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{02} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x \end{bmatrix} \quad T_3[x] = \begin{bmatrix} \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{03} \otimes_{\mathbb{F}_{2^8}} x \\ \mathbf{02} \otimes_{\mathbb{F}_{2^8}} x \end{bmatrix}$$

- **Idea:** pre-compute most of the round, i.e.,

MixColumns \circ ShiftRows \circ SubBytes.

- **Step #3:** ShiftRows and SubBytes can then be folded into the tables and look-ups, i.e.,

$$\begin{bmatrix} S'_{0,j} \\ S'_{1,j} \\ S'_{2,j} \\ S'_{3,j} \end{bmatrix} = T_0[s_{0,j+0 \pmod{Nr}}] \oplus_{\mathbb{F}_{2^8}} T_1[s_{1,j+1 \pmod{Nr}}] \oplus_{\mathbb{F}_{2^8}} T_2[s_{2,j+2 \pmod{Nr}}] \oplus_{\mathbb{F}_{2^8}} T_3[s_{3,j+3 \pmod{Nr}}]$$

where

$$T_0[x] = \begin{bmatrix} \mathbf{02} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{03} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \end{bmatrix} \quad T_1[x] = \begin{bmatrix} \mathbf{03} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{02} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \end{bmatrix}$$

$$T_2[x] = \begin{bmatrix} \mathbf{01} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{03} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{02} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \end{bmatrix} \quad T_3[x] = \begin{bmatrix} \mathbf{01} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{01} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{03} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \\ \mathbf{02} \otimes_{\mathbb{F}_{2^8}} \text{S-BOX}(x) \end{bmatrix}$$

Part 2: in practice (13)

Strategy #2

Listing

```
1 #define AES_ENC_RND_INIT() {           \  
2   t_0 = rkp[ 0 ] ^ s_0;               \  
3   t_1 = rkp[ 1 ] ^ s_1;               \  
4   t_2 = rkp[ 2 ] ^ s_2;               \  
5   t_3 = rkp[ 3 ] ^ s_3;               \  
6                                       \  
7   rkp += AES_128_NB; s_0 = t_0; s_1 = t_1; s_2 = t_2; s_3 = t_3;   \  
8 }
```

Listing

```

1 #define AES_ENC_RND_ITER() {
2   t_0 = rkp[ 0 ] ^ ( AES_ENC_TBOX_0[ ( s_0 >> 0 ) & 0xFF ] ) ^ \
3                   ( AES_ENC_TBOX_1[ ( s_1 >> 8 ) & 0xFF ] ) ^ \
4                   ( AES_ENC_TBOX_2[ ( s_2 >> 16 ) & 0xFF ] ) ^ \
5                   ( AES_ENC_TBOX_3[ ( s_3 >> 24 ) & 0xFF ] ) ; \
6   t_1 = rkp[ 1 ] ^ ( AES_ENC_TBOX_0[ ( s_1 >> 0 ) & 0xFF ] ) ^ \
7                   ( AES_ENC_TBOX_1[ ( s_2 >> 8 ) & 0xFF ] ) ^ \
8                   ( AES_ENC_TBOX_2[ ( s_3 >> 16 ) & 0xFF ] ) ^ \
9                   ( AES_ENC_TBOX_3[ ( s_0 >> 24 ) & 0xFF ] ) ; \
10  t_2 = rkp[ 2 ] ^ ( AES_ENC_TBOX_0[ ( s_2 >> 0 ) & 0xFF ] ) ^ \
11                  ( AES_ENC_TBOX_1[ ( s_3 >> 8 ) & 0xFF ] ) ^ \
12                  ( AES_ENC_TBOX_2[ ( s_0 >> 16 ) & 0xFF ] ) ^ \
13                  ( AES_ENC_TBOX_3[ ( s_1 >> 24 ) & 0xFF ] ) ; \
14  t_3 = rkp[ 3 ] ^ ( AES_ENC_TBOX_0[ ( s_3 >> 0 ) & 0xFF ] ) ^ \
15                  ( AES_ENC_TBOX_1[ ( s_0 >> 8 ) & 0xFF ] ) ^ \
16                  ( AES_ENC_TBOX_2[ ( s_1 >> 16 ) & 0xFF ] ) ^ \
17                  ( AES_ENC_TBOX_3[ ( s_2 >> 24 ) & 0xFF ] ) ; \
18
19  rkp += AES_128_NB; s_0 = t_0; s_1 = t_1; s_2 = t_2; s_3 = t_3;
20 }

```

Listing

```

1 #define AES_ENC_RND_FINI() {
2   t_0 = rkp[ 0 ] ^ ( AES_ENC_TBOX_4[ ( s_0 >> 0 ) & 0xFF ] & 0x000000FF ) ^ \
3                   ( AES_ENC_TBOX_4[ ( s_1 >> 8 ) & 0xFF ] & 0x0000FF00 ) ^ \
4                   ( AES_ENC_TBOX_4[ ( s_2 >> 16 ) & 0xFF ] & 0x00FF0000 ) ^ \
5                   ( AES_ENC_TBOX_4[ ( s_3 >> 24 ) & 0xFF ] & 0xFF000000 ) ; \
6   t_1 = rkp[ 1 ] ^ ( AES_ENC_TBOX_4[ ( s_1 >> 0 ) & 0xFF ] & 0x000000FF ) ^ \
7                   ( AES_ENC_TBOX_4[ ( s_2 >> 8 ) & 0xFF ] & 0x0000FF00 ) ^ \
8                   ( AES_ENC_TBOX_4[ ( s_3 >> 16 ) & 0xFF ] & 0x00FF0000 ) ^ \
9                   ( AES_ENC_TBOX_4[ ( s_0 >> 24 ) & 0xFF ] & 0xFF000000 ) ; \
10  t_2 = rkp[ 2 ] ^ ( AES_ENC_TBOX_4[ ( s_2 >> 0 ) & 0xFF ] & 0x000000FF ) ^ \
11                  ( AES_ENC_TBOX_4[ ( s_3 >> 8 ) & 0xFF ] & 0x0000FF00 ) ^ \
12                  ( AES_ENC_TBOX_4[ ( s_0 >> 16 ) & 0xFF ] & 0x00FF0000 ) ^ \
13                  ( AES_ENC_TBOX_4[ ( s_1 >> 24 ) & 0xFF ] & 0xFF000000 ) ; \
14  t_3 = rkp[ 3 ] ^ ( AES_ENC_TBOX_4[ ( s_3 >> 0 ) & 0xFF ] & 0x000000FF ) ^ \
15                  ( AES_ENC_TBOX_4[ ( s_0 >> 8 ) & 0xFF ] & 0x0000FF00 ) ^ \
16                  ( AES_ENC_TBOX_4[ ( s_1 >> 16 ) & 0xFF ] & 0x00FF0000 ) ^ \
17                  ( AES_ENC_TBOX_4[ ( s_2 >> 24 ) & 0xFF ] & 0xFF000000 ) ; \
18
19  rkp += AES_128_NB; s_0 = t_0; s_1 = t_1; s_2 = t_2; s_3 = t_3;
20 }

```

Listing

```
1 void aes_enc( uint8_t* c, const uint8_t* m, const uint8_t* rk ) {
2   uint32_t s_0, s_1, s_2, s_3, t_0, t_1, t_2, t_3;
3
4   U8_TO_U32_LE( s_0, m, 0 ); U8_TO_U32_LE( s_1, m, 4 );
5   U8_TO_U32_LE( s_2, m, 8 ); U8_TO_U32_LE( s_3, m, 12 );
6
7   uint32_t *rkp = ( uint32_t* )( rk );
8
9   //      1 initial round
10  AES_ENC_RND_INIT();
11  // Nr - 1 iterated rounds
12  for( int i = 1; i < AES_128_NR; i++ ) {
13    AES_ENC_RND_ITER();
14  }
15  //      1 final round
16  AES_ENC_RND_FINI();
17
18  U32_TO_U8_LE( c, s_0, 0 ); U32_TO_U8_LE( c, s_1, 4 );
19  U32_TO_U8_LE( c, s_2, 8 ); U32_TO_U8_LE( c, s_3, 12 );
20 }
```

Part 2: in practice (15)

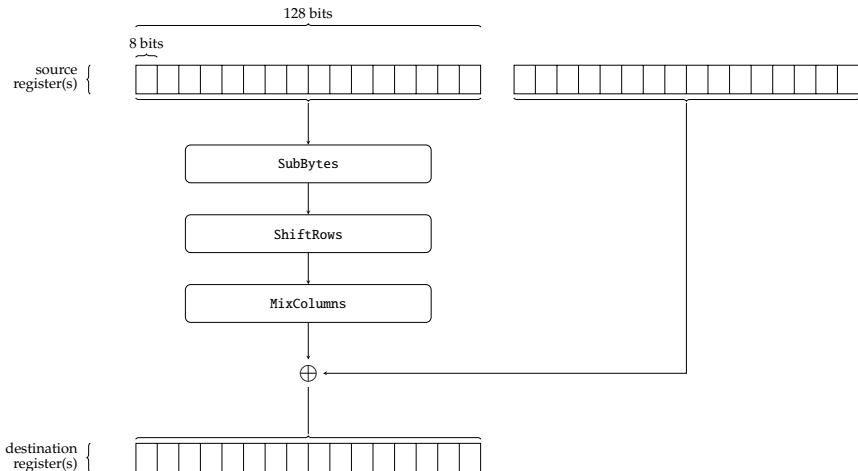
Strategy #3

- ▶ **Strategy #3 [7]** (i.e., Intel AES-NI):
 - ▶ use hybrid of software (i.e., via ISA) and hardware (i.e., via ISE),
 - ▶ adopt a fully-packed representation of state and round key matrices, using 128-bit words, i.e., instances of type `__m128i`,
 - ▶ favour use of straight-line (i.e., unrolled) loops,
 - ▶ pre-compute (or expand) round keys,
 - ▶ compute round functions.

Part 2: in practice (16)

Strategy #3

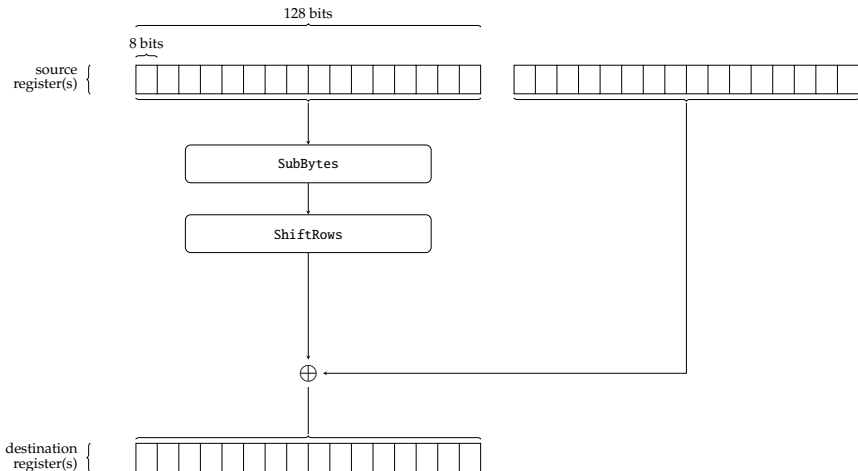
Circuit (aesenc [6, Pages 3-54–3-55])



Part 2: in practice (16)

Strategy #3

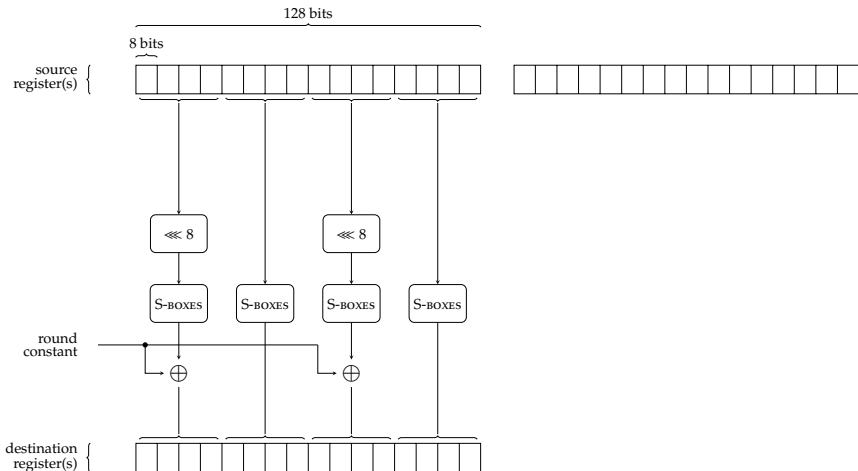
Circuit (aesenc1ast [6, Pages 3-56–3-57])



Part 2: in practice (16)

Strategy #3

Circuit (aeskeygenassist [6, Pages 3-59-3-60])



Listing

```
1 void aes_enc( uint8_t* c, const uint8_t* m, const uint8_t* rk ) {
2   __m128i s = _mm_load_si128( ( __m128i* )( m ) );
3   __m128i* rkp = ( __m128i* )( rk );
4
5   //   1 initial round
6   s =   _mm_xor_si128( s, _mm_load_si128( rkp++ ) );
7   // Nr - 1 iterated rounds
8   s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
9   s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
10  s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
11  s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
12  s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
13  s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
14  s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
15  s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
16  s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
17  //   1 final round
18  s =   _mm_aesenc_si128( s, _mm_load_si128( rkp++ ) );
19
20  _mm_store_si128( ( __m128i* )( c ), s );
21 }
```

Foot-Shooting Prevention Agreement

I, _____, promise that once
Your Name

I see how simple AES really is, I will not implement it in production code even though it would be really fun.

This agreement shall be in effect until the undersigned creates a meaningful interpretive dance that compares and contrasts cache-based, timing, and other side channel attacks and their countermeasures.

X _____
Signature Date

- ▶ **Take away points:** you can often simply *use*

$$c = \text{AES-128.ENC}(k, m),$$

but understanding internals of this primitive can be useful and/or important.

- ▶ some historically interesting aspects; some “portable” concepts,
- ▶ close relationship between primitive and underlying Mathematics,
- ▶ wide range of viable implementation strategies,
- ▶ extensive deployment, in various contexts and use-cases.

Additional Reading

- ▶ *Wikipedia: Advanced Encryption Standard (AES)*. URL: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard.
- ▶ *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology (NIST) Federal Information Processing Standard (FIPS) 197 (update 1). 2023. URL: <http://csrc.nist.gov>.
- ▶ L.R. Knudsen and M.J.B. Robshaw. *The Block Cipher Companion*. Springer, 2011.
- ▶ J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer, 2002.

References

- [1] *Wikipedia: Advanced Encryption Standard (AES)*. URL: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard (see p. 45).
- [2] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer, 2002 (see pp. 1, 20, 27, 45).
- [3] L.R. Knudsen and M.J.B. Robshaw. *The Block Cipher Companion*. Springer, 2011 (see p. 45).
- [4] J. Daemen, L. Knudsen, and V. Rijmen. “The block cipher Square”. In: *Fast Software Encryption (FSE)*. LNCS 1267. Springer-Verlag, 1997, pp. 149–165 (see p. 1).
- [5] J. Daemen and V. Rijmen. “The Block Cipher Rijndael”. In: *Smart Card Research and Applications (CARDIS)*. LNCS 1820. Springer-Verlag, 1998, pp. 277–284 (see p. 1).
- [6] *Intel 64 and IA-32 architectures – Software Developer’s Manual (Volume 2: Instruction Set Reference A-Z)*. Tech. rep. 325383-071US. Intel Corp., 2019. URL: <http://software.intel.com/en-us/articles/intel-sdm> (see pp. 39–41).
- [7] *Intel Advanced Encryption Standard (AES) Instructions Set*. Tech. rep. Intel Corp., 2012. URL: <http://software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf> (see p. 38).
- [8] *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology (NIST) Federal Information Processing Standard (FIPS) 197 (update 1). 2023. URL: <http://csrc.nist.gov> (see pp. 1, 3, 4, 8–18, 45).