1997 need to replace DES for encrypting goverment documents.
The new algo. would be called AES. Risnadael was chosen

The finite body of Risndael: $\mathbb{F}_{256}$

The algorithm uses bytes, so it's usefull to use a body with 256 elements
We will use the polinomial:

$$g(x) = x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[X]$$

which is irreducible over $\mathbb{F}_2$ and we will therefore call $\mathbb{F}_{256}$ the body of 256 el. of the form:

$$GF(2^8) = \mathbb{F}_{256} = \mathbb{F}_2[X]/(g(x))$$

we note ξ as the generetor of the multiplicative group of this body. We get the conversion
$\{bytes\} \to \mathbb{F}_{256} := (a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0)_2 \to a_7 x^7 \dots a_1 x + a_0 \pmod{g(x)}$
e.g. $10011011 \to x^7 + x^4 + x^3 + x + 1$. We could also use Hex for representation
We implement addition via bitwise XOR. For add/sub we don't need the complicated struct of the field
For multypling bytes we first need to convert them to pols., do the multiplication modulo g(x)

example:
calculate $9B \cdot 88$ in $GF(2^8)$
treat numbers as polynomials
then it becomes $10011011 \cdot 10001000$
all modulo $x^8+x^4+x^3+x+1$

$$(x^7+x^4+x^3+x+1)(x^7+x^3) \equiv (x^{14}+x^{11}+x^{10}+x^8+x^7)+(x^{10}+x^7+x^6+x^4+x^3) \equiv$$
$$\equiv x^{14}+x^{11}+x^8+x^6+x^4+x^3 \equiv 1$$

the last part can be done by working out a long division and finding the rest

```
100100101011000      | 100011011
100011011            |-----------
---------
   111110011000
   100011011
   ---------
    11101000000
    100011011
    ---------
     1100101100
     100011011
     ---------
      100011010
      100011011
      ---------
              1
```

there's a more efficient way to do it

We note $\xi = \{03\} = x + 1$ is a primitive element of the body (see later part)
Every element of the body except 0 can be written as a power of this generator.
Because of this we can use some convertion tables

let's do 9B·88 again

L-table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		00	19	01	32	02	1A	C6	4B	C7	1B	68	33	EE	DF	03
1	64	04	E0	0E	34	8D	81	EF	4C	71	08	C8	F8	69	1C	C1
2	7D	C2	1D	B5	F9	B9	27	6A	4D	E4	A6	72	9A	C9	09	78
3	65	2F	8A	05	21	0F	E1	24	12	F0	82	45	35	93	DA	8E
4	96	8F	DB	BD	36	D0	CE	94	13	5C	D2	F1	40	46	83	38
5	66	DD	FD	30	BF	06	8B	62	B3	25	E2	98	22	88	91	10
6	7E	6E	48	C3	A3	B6	1E	42	3A	6B	28	54	FA	85	3D	BA
7	2B	79	0A	15	9B	9F	5E	CA	4E	D4	AC	E5	F3	73	A7	57
8	AF	58	A8	50	F4	EA	D6	74	4F	AE	E9	D5	E7	E6	AD	E8
9	2C	D7	75	7A	EB	16	0B	F5	59	CB	5F	B0	9C	A9	51	A0
A	7F	0C	F6	6F	17	C4	49	EC	D8	43	1F	2D	A4	76	7B	B7
B	CC	BB	3E	5A	FB	60	B1	86	3B	52	A1	6C	AA	55	29	9D
C	97	B2	87	90	61	BE	DC	FC	BC	95	CF	CD	37	3F	5B	D1
D	53	39	84	3C	41	A2	6D	47	14	2A	9E	5D	56	F2	D3	AB
E	44	11	92	D9	23	20	2E	89	B4	7C	B8	26	77	99	E3	A5
F	67	4A	ED	DE	C5	31	FE	18	0D	63	8C	80	C0	F7	70	07

$9B = \xi_7^{B0}$

$88 = \xi_7^{4F}$

$\xi_7^{B0} \cdot \xi_7^{4F} = \xi_7^{FF}$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	01	03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35
1	5F	E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA
2	E5	34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31
3	53	F5	04	0C	14	3C	44	CC	4F	D1	68	B8	D3	6E	B2	CD
4	4C	D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88
5	83	9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A
6	B5	C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3
7	FE	19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0
8	FB	16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41
9	C3	5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75
A	9F	BA	D5	64	AC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80
B	9B	B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54
C	FC	1F	21	63	A5	F4	07	09	1B	2D	77	99	B0	CB	46	CA
D	45	CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E
E	12	36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17
F	39	4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01

like before

$\xi_7^{FF} = 01$

if you get a power > FF you subtract FF

as a generator of $GF(2^8)$ it was decided to take $\xi_{03} = x+4$.
this element has for minimum term over $\mathbb{F}_2$ the poly. $x^8+x^4+x^3+x^2+1 \neq g(x)$

Working of Rijndael

The text is divided into blocks of 128 bits (or 192 or 256)
this results in a 4x4 (or 4x6 or 4x8) matrix of bytes. We'll use the following notation

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

each a_{ij} represents 1 Byte.

This matrix is adjusted in a number of rounds to sum the entries (XOR) with a key, replace them with other elements of $\mathbb{F}_{256}$ and permute

So a key is used (128 b) from which an expanded key is created that has the same length of the block (128) multiplied by the number of rounds +1.
In our example 10 steps (standard for 128b keys). Then the expanded key has length $11 \cdot 128$
we will call it $S_0, S_1 \dots S_{10}$

In big steps:

- Before starting add S_0 to the text
- 9 round of 4 steps
    1) SubBytes : a substitution of matrix entries
    2) Shiftrow : permutation of matrix entries
    3) MixColums : matrix multiplication
    4) AddRoundKey: S_i added
- with i=10 ③ is skipped

1) SubBytes

Replace each Byte a_{ij} with b_{ij}

$a \to b$ is called S-box and is a function $g: \mathbb{F}_{256} \to \mathbb{F}_{256} : a \to c = \begin{cases} a^{-1} & \text{if } a \neq 0 \\ 0 & \text{if } a = 0 \end{cases}$

followed by $f: \mathbb{F}_{256} \to \mathbb{F}_{256} : c \to b$ where $b = b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$ is the result of

S-box = $f \circ g$ can also be summarized in a table

$$\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

two hex → 1 Byte

2) Shiftrows

entries are permuted, first row unchanged, the other are shifted by i to the left

3) MixColumns

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \to \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

4) AddRoundKey

the key is added

XOR

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \to \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{22} & a_{23} & a_{24} & a_{21} \\ a_{33} & a_{34} & a_{31} & a_{32} \\ a_{44} & a_{41} & a_{42} & a_{43} \end{pmatrix} + \begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{22} & k_{23} & k_{24} & k_{21} \\ k_{33} & k_{34} & k_{31} & k_{32} \\ k_{44} & k_{41} & k_{42} & k_{43} \end{pmatrix}$$

Decryption

Same scheme backwards

- addroundkey^{-1} : add & sub same in mod 2
- mixcolumn^{-1} : multiply by different matrix

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \to \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

- shiftrow^{-1} : shift to right instead of left
- subBytes: we have to set up an analoge table for the inverse