

## 8. Šifrovací algoritmus AES

Jan Paseka

Ústav matematiky a statistiky  
Masarykova univerzita

6. prosince 2021

# O čem to bude



- 1 Potřeba a historie vzniku AES
  - Potřeba AES
- 2 Popis šifrovacího algoritmu AES

- 3 Tvorba klíčů
- 4 Šifrovací proces
- 5 Dešifrovací proces

# Úvod I

**FI** V současné době je otázka šifrování velice důležitou nejen z hlediska bezpečnosti a ochrany, ale také pro další rozvoj informačních technologií. Šifrování a další kryptografické postupy jsou v mnoha případech jedinou skutečně účinnou metodou, jak data chránit.

# Úvod I

**FI** V současné době je otázka šifrování velice důležitou nejen z hlediska bezpečnosti a ochrany, ale také pro další rozvoj informačních technologií. Šifrování a další kryptografické postupy jsou v mnoha případech jedinou skutečně účinnou metodou, jak data chránit.

Šifrovací algoritmus AES (Advanced Encryption Standard) je současným průmyslovým šifrovacím standardem. Šifra AES nahradila dříve používanou šifru DES.

# Úvod I

**FI** V současné době je otázka šifrování velice důležitou nejen z hlediska bezpečnosti a ochrany, ale také pro další rozvoj informačních technologií. Šifrování a další kryptografické postupy jsou v mnoha případech jedinou skutečně účinnou metodou, jak data chránit.

Šifrovací algoritmus AES (Advanced Encryption Standard) je současným průmyslovým šifrovacím standardem. Šifra AES nahradila dříve používanou šifru DES.

Její původní název je Rijndael. Název vznikl přesmyčkou jmen jejích dvou belgických autorů Vincenta Rijnmanna a Joana Daemena.

# Úvod I

**FI** V současné době je otázka šifrování velice důležitou nejen z hlediska bezpečnosti a ochrany, ale také pro další rozvoj informačních technologií. Šifrování a další kryptografické postupy jsou v mnoha případech jedinou skutečně účinnou metodou, jak data chránit.

Šifrovací algoritmus AES (Advanced Encryption Standard) je současným průmyslovým šifrovacím standardem. Šifra AES nahradila dříve používanou šifru DES.

Její původní název je Rijndael. Název vznikl přesmyčkou jmen jejich dvou belgických autorů Vincenta Rijnmanna a Joana Daemena.

Autoři svoji šifru přihlásili 2. ledna 1997 na veřejnou soutěž o federální šifrovací algoritmus pořádanou Národním institutem standardů a technologii (NIST).

# Úvod II

Po pěti letech byla šifra AES schválena jako nejvhodnější z patnácti návrhů. Dne 26. května 2002 začala být ke svému účelu používána jako federální standard USA. AES je první šifra, dostupná široké veřejnosti, která je zároveň uznána Národní bezpečností agenturou NSA.

# Úvod II

Po pěti letech byla šifra AES schválena jako nejvhodnější z patnácti návrhů. Dne 26. května 2002 začala být ke svému účelu používána jako federální standard USA. AES je první šifra, dostupná široké veřejnosti, která je zároveň uznána Národní bezpečností agenturou NSA.

V současnosti se využívá k šifrování elektronické pošty, elektronického bankovníctví, různých druhů dálkové autentizace, čipových karet, elektronických peněz, přenosu hovorů v síti GSM, signálu wi-fi, bluetooth a satelitů.



# O čem to bude



- 1 Potřeba a historie vzniku AES
- 2 Popis šifrovacího algoritmu AES
  - Bloková šifra
- 3 Tvorba klíčů
- 4 Šifrovací proces
- 5 Dešifrovací proces

# Bloková šifra I

AES je symetrická šifra, využívající bloky dat velké **128 bitů** a klíče velikosti **128, 192** nebo **256** bitů.

# Bloková šifra I

AES je symetrická šifra, využívající bloky dat velké **128 bitů** a klíče velikosti **128, 192** nebo **256** bitů.

Velikost klíče ovlivňuje počet rund (může jich být **10, 12** nebo **14**).

# Bloková šifra I

AES je symetrická šifra, využívající bloky dat velké **128 bitů** a klíče velikosti **128, 192** nebo **256** bitů.

Velikost klíče ovlivňuje počet rund (může jich být **10, 12** nebo **14**).

Pro grafickou jednoduchost data uspořádáváme do matice  $4 \times 4$  označované jako **State** (stav). Data jsou do matice vkládána po sloupcích zprava doleva.

# Bloková šifra I

AES je symetrická šifra, využívající bloky dat velké **128 bitů** a klíče velikosti **128, 192** nebo **256** bitů.

Velikost klíče ovlivňuje počet rund (může jich být **10, 12** nebo **14**).

Pro grafickou jednoduchost data uspořádáváme do matice  $4 \times 4$  označované jako **State** (stav). Data jsou do matice vkládána po sloupcích zprava doleva.

Prvky stavové matice (byty zapsané v hexadecimálním tvaru) jsou chápány jako polynomy a pracuje se s nimi modulo  $m(x) = x^8 + x^4 + x^3 + x^1 + 1$ .

# Bloková šifra I

AES je symetrická šifra, využívající bloky dat velké **128 bitů** a klíče velikosti **128, 192** nebo **256** bitů.

Velikost klíče ovlivňuje počet rund (může jich být **10, 12** nebo **14**).

Pro grafickou jednoduchost data uspořádáváme do matice  $4 \times 4$  označované jako **State** (stav). Data jsou do matice vkládána po sloupcích zprava doleva.

Prvky stavové matice (byty zapsané v hexadecimálním tvaru) jsou chápány jako polynomy a pracuje se s nimi modulo  $m(x) = x^8 + x^4 + x^3 + x^1 + 1$ . Platí tedy, že

$$GF(2^8) = GF(2)(\alpha),$$

přičemž  $\alpha$  splňuje rovnici

$$\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1 = 0.$$

# Bloková šifra III

Speciálně jeden byte  
prvku

$(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$

odpovídá

$$\sum_{i=0}^7 b_i \alpha^i.$$

# Bloková šifra III

Speciálně jeden byte  
prvku

$(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$

odpovídá

$$\sum_{i=0}^7 b_i \alpha^i.$$

Tímto způsobem **můžeme byty násobit a sčítat**. Pokud jsou nenulové, mají vždy inverzi, kterou pro byte  $b$  budeme značit  $b^{-1}$ . Zároveň položíme  $0^{-1} = 0$ .



# Bloková šifra III

Speciálně jeden byte  
prvku

$$(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$$

odpovídá

$$\sum_{i=0}^7 b_i \alpha^i.$$

Tímto způsobem **můžeme byty násobit a sčítat**. Pokud jsou nenulové, mají vždy inverzi, kterou pro byte  $b$  budeme značit  $b^{-1}$ . Zároveň položíme  $0^{-1} = 0$ .

Algoritmus se dělí na dva procesy, samotné šifrování a tvorbu klíčů. K šifrování jsou využívány čtyři procedury **SubBytes**, **ShiftRows**, **MixColumns** a **AddRoundKey**. Pro dešifrování jsou poté použity procedury inverzní.

# Bloková šifra III

Speciálně jeden byte  
prvku

$$(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$$

odpovídá

$$\sum_{i=0}^7 b_i \alpha^i.$$

Tímto způsobem **můžeme byty násobit a sčítat**. Pokud jsou nenulové, mají vždy inverzi, kterou pro byte  $b$  budeme značit  $b^{-1}$ . Zároveň položíme  $0^{-1} = 0$ .

Algoritmus se dělí na dva procesy, samotné šifrování a tvorbu klíčů. K šifrování jsou využívány čtyři procedury **SubBytes**, **ShiftRows**, **MixColumns** a **AddRoundKey**. Pro dešifrování jsou poté použity procedury inverzní.

Tvorba klíčů slouží k vytvoření nových klíčů z původního zadaného klíče.

# O čem to bude



- 1 Potřeba a historie vzniku AES
- 2 Popis šifrovacího algoritmu AES

- 3 Tvorba klíčů
  - Tvorba nových klíčů
- 4 Šifrovací proces
- 5 Dešifrovací proces

# Tvorba nových klíčů I

Šifrovací klíč zadáváme na začátku, před započítím šifrovacího procesu. Může být různě dlouhý. Na jeho délce závisí počet provedených rund.

# Tvorba nových klíčů I

Šifrovací klíč zadáváme na začátku, před započítím šifrovacího procesu. Může být různě dlouhý. Na jeho délce závisí počet provedených rund.

Kratší šifrovací klíč nám zajistí úsporu času, potřebného pro provedení šifrování. Delší klíč nám zvýší bezpečnost šifry, ale na úkor rychlosti.

# Tvorba nových klíčů I

Šifrovací klíč zadáváme na začátku, před započítím šifrovacího procesu. Může být různě dlouhý. Na jeho délce závisí počet provedených rund.

Kratší šifrovací klíč nám zajistí úsporu času, potřebného pro provedení šifrování. Delší klíč nám zvýší bezpečnost šifry, ale na úkor rychlosti.

Klíč zapisujeme do matice o rozměrech  $N \times 4$ , kde  $N$  je rovno **4**, **6** nebo **8** pro klíč dlouhý **128**, **196** nebo **256** bitů.

# Tvorba nových klíčů I

Šifrovací klíč zadáváme na začátku, před započítím šifrovacího procesu. Může být různě dlouhý. Na jeho délce závisí počet provedených rund.

Kratší šifrovací klíč nám zajistí úsporu času, potřebného pro provedení šifrování. Delší klíč nám zvýší bezpečnost šifry, ale na úkor rychlosti.

Klíč zapisujeme do matice o rozměrech  $N \times 4$ , kde  $N$  je rovno **4**, **6** nebo **8** pro klíč dlouhý **128**, **196** nebo **256** bitů.

Pro vytvoření nových klíčů využijeme proceduru **KeyExpansion**. Nový klíč v prvním kroku vytvoříme z námi zadaného klíče. Další klíče se vytvářejí pomocí nově vytvořených klíčů.

## Tvorba nových klíčů II

Na začátku procesu tvorby klíče vezmeme ***poslední sloupek našeho klíče***.



## Tvorba nových klíčů II

Na začátku procesu tvorby klíče vezmeme ***poslední sloupek našeho klíče***.

Prvky posuneme o jedna nahoru (první prvek přesuneme na spodek sloupce), všechny prvky sloupce nahradíme pomocí procedury ***SubBytes*** (více o proceduře SubBytes v části "Šifrovací proces").

## Tvorba nových klíčů II

Na začátku procesu tvorby klíče vezmeme ***poslední sloupek našeho klíče***.

Prvky posuneme o jedna nahoru (první prvek přesuneme na spodek sloupce), všechny prvky sloupce nahradíme pomocí procedury ***SubBytes*** (více o proceduře SubBytes v části "Šifrovací proces").

Operací ***Xor sečteme se sloupcem o indexu pole o 3 menší než je index našeho sloupce. Vzniklý sloupec ještě sečteme s příslušným sloupcem tabulky RCon.***

## Tvorba nových klíčů II

Na začátku procesu tvorby klíče vezmeme ***poslední sloupek našeho klíče***.

Prvky posuneme o jedna nahoru (první prvek přesuneme na spodek sloupce), všechny prvky sloupce nahradíme pomocí procedury ***SubBytes*** (více o proceduře SubBytes v části "Šifrovací proces").

Operací ***Xor sečteme se sloupcem o indexu pole o 3 menší než je index našeho sloupce. Vzniklý sloupec ještě sečteme s příslušným sloupcem tabulky RCon.***

Další tři sloupce spočítáme sečtením vždy nově vytvořeného sloupce a sloupce o indexu pole o 3 menším. Tento proces několikrát opakujeme.

## Tvorba nových klíčů III

Námi vytvořené klíče se později přičítají s poli State v průběhu procedury AddRoundKey (více o proceduře AddRoundKey v části "Šifrovací proces")

# Tvorba nových klíčů III

Námi vytvořené klíče se později přičítají s poli State v průběhu procedury AddRoundKey (více o proceduře AddRoundKey v části "Šifrovací proces")

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
01	02	04	08	10	20	40	80	1B	36	6C	D8	AB	4D	9A	2F
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Tabulka 1: RCon

# O čem to bude

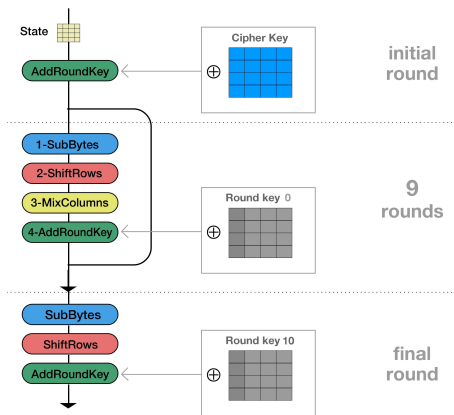


- 1 Potřeba a historie vzniku AES
- 2 Popis šifrovacího algoritmu AES
- 3 Tvorba klíčů

- 4 Šifrovací proces
  - Čtyři procedury
  - Procedura SubBytes
  - Procedura ShiftRows
  - Procedura MixColumns
  - Procedura AddRoundKey
- 5 Dešifrovací proces

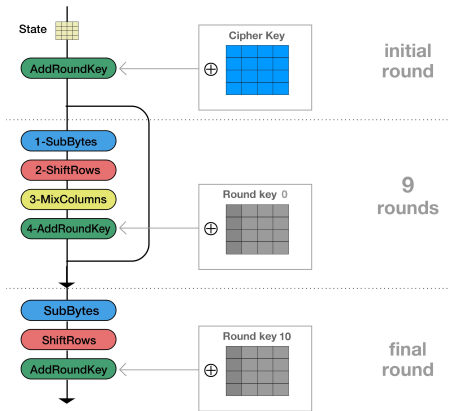
# Čtyři procedury I

Jak už bylo řečeno, k šifrovacímu procesu slouží čtyři procedury, které upraví zadaný vstup.



# Čtyři procedury II

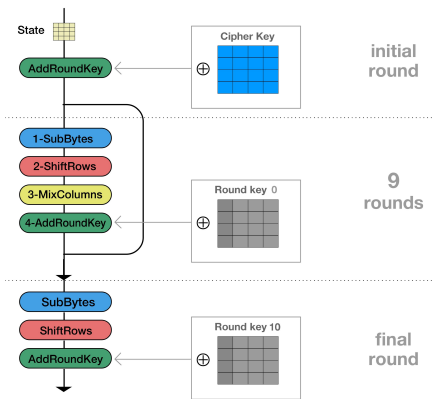
Na úvod se State (naš vstup) sečte operací Xor s námi zadaným klíčem.





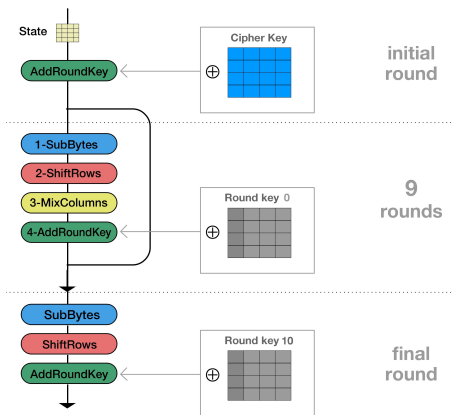
# Čtyři procedury III

Následuje 10-14 rund v závislosti na velikosti klíče. Během nich se vždy zopakují procedury SubBytes, ShiftRows, MixColumns a AddRoundKey.



# Čtyři procedury IV

Vyjimku tvoří poslední runda, v níž se procedura MixColumns vynechává.



# Procedura SubBytes I

Procedura SubBytes je **nelineární funkce**, která transformuje jednotlivé byty pole State. Tato transformace se nazývá S-Box. Z každého bytu  $b = (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)^T$  pole State vytvoří S-Box nový byte  $\bar{b}$  předpisem

$$\bar{b} = \mathbf{A}b^{-1} + c,$$

kde

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad c = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

# Procedura SubBytes II

## Příklad 4.1

Byte  $b = (0, 0, 0, 0, 0, 0, 1, 1)^T$  odpovídá prvku tělesa tvaru  $\alpha + 1$  a v hexadecimální notaci má zápis **03**.

# Procedura SubBytes II

## Příklad 4.1

Byte  $b = (0, 0, 0, 0, 0, 0, 1, 1)^T$  odpovídá prvku tělesa tvaru  $\alpha + 1$  a v hexadecimální notaci má zápis **03**.

Snadno se ověří, že

$$(\alpha + 1)^{-1} = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha.$$

# Procedura SubBytes II

## Příklad 4.1

Byte  $b = (0, 0, 0, 0, 0, 0, 1, 1)^T$  odpovídá prvku tělesa tvaru  $\alpha + 1$  a v hexadecimální notaci má zápis **03**.

Snadno se ověří, že

$$(\alpha + 1)^{-1} = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha.$$

Tedy pak

$$b^{-1} = (1, 1, 1, 1, 0, 1, 1, 0)^T$$

a v hexadecimální notaci má  $b^{-1}$  zápis **F6**.

# Procedura SubBytes II

## Příklad 4.1

Byte  $b = (0, 0, 0, 0, 0, 0, 1, 1)^T$  odpovídá prvku tělesa tvaru  $\alpha + 1$  a v hexadecimální notaci má zápis **03**.

Snadno se ověří, že

$$(\alpha + 1)^{-1} = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha.$$

Tedy pak

$$b^{-1} = (1, 1, 1, 1, 0, 1, 1, 0)^T$$

a v hexadecimální notaci má  $b^{-1}$  zápis **F6**. Zejména pak

$$\bar{b} = Ab^{-1} + c = (0, 1, 1, 0, 0, 1, 1, 1)^T$$

# Procedura SubBytes II

## Příklad 4.1

Byte  $b = (0, 0, 0, 0, 0, 0, 1, 1)^T$  odpovídá prvku tělesa tvaru  $\alpha + 1$  a v hexadecimální notaci má zápis **03**.

Snadno se ověří, že

$$(\alpha + 1)^{-1} = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha.$$

Tedy pak

$$b^{-1} = (1, 1, 1, 1, 0, 1, 1, 0)^T$$

a v hexadecimální notaci má  $b^{-1}$  zápis **F6**. Zejména pak

$$\bar{b} = Ab^{-1} + c = (0, 1, 1, 0, 0, 1, 1, 1)^T$$

a odpovídající zápis v hexadecimální notaci je **67**.



# Procedura SubBytes III - substituční tabulka S-Box

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

# Procedura SubBytes IV - substituční tabulka S-Box

hex		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	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	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	5	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	6	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	7	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	8	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	9	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	A	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	B	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	C	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	D	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	E	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16
	F																

Má 16 řádků a 16 sloupců.

# Procedura SubBytes IV - substituční tabulka S-Box

hex	y															
	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	53	32	39	7A	48	34	4A	5A	A0	52	3B	D6	B3	29	E3	2F
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	50	EF	AA	7B	48	4D	53	85	46	7F	01	77	3D	5C	14	A8
7	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

Má 16 řádků a 16 sloupců.

Každý prvek tabulky má jeden bajt (stejně jako každý prvek pole State).

# Procedura SubBytes IV - substituční tabulka S-Box

hex	y															
	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	8E	41	8D	52	9E	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	9D	50	EF	AA	7B	48	4D	83	46	1F	01	7F	50	3C	4F	A8
7	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	BC	6C	4F	DD	2E	2A	90	89	40	22	B8	14	DE	3E	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

Má 16 řádků a 16 sloupců.

Každý prvek tabulky má jeden bajt (stejně jako každý prvek pole State).

Všechny prvky pole State nahradíme pomocí S-Boxu.

# Procedura SubBytes IV - substituční tabulka S-Box

hex	y															
	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	84	1A	CC	5E	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	90	EB	AA	1B	43	4D	53	83	46	71	01	77	39	5C	4F	A8
7	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	40	EE	D8	14	DE	3E	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

Má 16 řádků a 16 sloupců.

Každý prvek tabulky má jeden bajt (stejně jako každý prvek pole State).

Všechny prvky pole State nahradíme pomocí S-Boxu.

První čtyři bity prvku označují řádek v tabulce S-Box, další čtyři sloupec.

# Procedura SubBytes IV - substituční tabulka S-Box

hex	y															
	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	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	D9	EF	AA	FB	43	4D	53	83	4B	F9	01	77	59	3C	4F	A8
7	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	BC	42	68	DC	22	2A	90	88	40	EE	D8	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	BF	C6	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	33	41	69	D3	8E	94	9B	1E	87	E9	CE	50	E3	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Má 16 řádků a 16 sloupců.

Každý prvek tabulky má jeden bajt (stejně jako každý prvek pole State).

Všechny prvky pole State nahradíme pomocí S-Boxu.

První čtyři bity prvku označují řádek v tabulce S-Box, další čtyři sloupec.

Například prvek **19** ukazuje v S-Boxu na hodnotu **D4**.

# Procedura SubBytes V

59	44	8B	29
11	80	D7	EF
63	DF	35	6F
77	AC	A8	1C

⇒

CB	1B	3D	A5
82	CD	0E	DF
FB	9E	96	A8
F5	91	C2	9C

Tabulka 2: Procedura SubBytes

hex	y															
	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

# Procedura ShiftRows I

Při proceduře ShiftRows dojde k posunu v rámci řádků v poli State. První řádek zůstává stejný, druhý řádek se posune o jednu pozici doleva, třetí řádek o dvě pozice a čtvrtý řádek o tři pozice.



# Procedura ShiftRows I

Při proceduře ShiftRows dojde k posunu v rámci řádků v poli State. První řádek zůstává stejný, druhý řádek se posune o jednu pozici doleva, třetí řádek o dvě pozice a čtvrtý řádek o tři pozice.

$$\begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{pmatrix} \leftarrow \begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{pmatrix}$$

# Procedura ShiftRows I

Při proceduře ShiftRows dojde k posunu v rámci řádků v poli State. První řádek zůstává stejný, druhý řádek se posune o jednu pozici doleva, třetí řádek o dvě pozice a čtvrtý řádek o tři pozice.

$$\begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{pmatrix} \leftarrow \begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{pmatrix}$$

Tato procedura nám při jejím použití ve více rundách zajistí **vysokou difúzi**.

# Procedura ShiftRows II

Při proceduře ShiftRows dojde k posunu v rámci řádků v poli State. První řádek zůstává stejný, druhý řádek se posune o jednu pozici doleva, třetí řádek o dvě pozice a čtvrtý řádek o tři pozice.

# Procedura ShiftRows II

Při proceduře ShiftRows dojde k posunu v rámci řádků v poli State. První řádek zůstává stejný, druhý řádek se posune o jednu pozici doleva, třetí řádek o dvě pozice a čtvrtý řádek o tři pozice.

CB	1B	3D	A5			CB	1B	3D	A5	
82	CD	0E	DF	←		CD	0E	DF	82	
FB	9E	96	A8	←	←	96	A8	FB	9E	
F5	91	C2	9C	←	←	←	9C	F5	91	C2

Tabulka 3: Procedura ShiftRows

# Procedura MixColumns I

V proceduře MixColumns dojde ke změně jednotlivých sloupců pole State. Každý bajt se změní na novou hodnotu, která je funkcí všech čtyř bajtů sloupce. Operace se provádí modulo  $m(x) = x^8 + x^4 + x^3 + x^1 + 1$ .

# Procedura MixColumns I

V proceduře MixColumns dojde ke změně jednotlivých sloupců pole State. Každý bajt se změní na novou hodnotu, která je funkcí všech čtyř bajtů sloupce. Operace se provádí modulo  $m(x) = x^8 + x^4 + x^3 + x^1 + 1$ .

Přesněji sloupec  $s_j = (s_{0,j}, s_{1,j}, s_{2,j}, s_{3,j})^T$  pole State budeme identifikovat s polynomem

$$s_j(x) = s_{0,j} + s_{1,j}x + s_{2,j}x^2 + s_{3,j}x^3 \in GF(2^8)[x].$$

# Procedura MixColumns I

V proceduře MixColumns dojde ke změně jednotlivých sloupců pole State. Každý bajt se změní na novou hodnotu, která je funkcí všech čtyř bajtů sloupce. Operace se provádí modulo  $m(x) = x^8 + x^4 + x^3 + x^1 + 1$ .

Přesněji sloupec  $s_j = (s_{0,j}, s_{1,j}, s_{2,j}, s_{3,j})^T$  pole State budeme identifikovat s polynomem

$$s_j(x) = s_{0,j} + s_{1,j}x + s_{2,j}x^2 + s_{3,j}x^3 \in GF(2^8)[x].$$

Procedura MixColumns převede sloupec  $s_j$  na sloupec  $\bar{s}_j$  tak, že

$$\bar{s}_j(x) = (s_j(x) * a(x)) \bmod (x^4 + 1),$$

přičemž

$$a(x) = 03 * x^3 + 01 * x^2 + 01 * x + 02.$$

# Procedura MixColumns II

Jedná se o **lineární proceduru** v  $GF(2^8)[x]$ , která provádí zajištění vysoké difúze ve sloupcích stavové matice.



# Procedura MixColumns II

Jedná se o **lineární proceduru** v  $GF(2^8)[x]$ , která provádí zajištění vysoké difúze ve sloupcích stavové matice.

CB	1B	3D	A5
CD	0E	DF	82
96	A8	FB	9E
9C	F5	91	C2

$$\Rightarrow \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} \text{CB} \\ \text{CD} \\ 96 \\ 9C \end{bmatrix} = \begin{bmatrix} \text{CB} \\ 77 \\ 8E \\ 3E \end{bmatrix}$$

Tabulka 4: Procedura MixColumns

Tato procedura zajistí v **každém kole závislost na klíči**.

# Procedura AddRoundKey I

Jednotlivé prvky pole State se přičtou pomocí operace Xor ke stejným prvkům příslušného rundovního klíče.

# Procedura AddRoundKey I

Jednotlivé prvky pole State se přičtou pomocí operace Xor ke stejným prvkům příslušného rundovního klíče.

CB	79	6A	90	$\oplus$	A0	88	23	2A	=	6B	F1	49	BA
77	11	1F	C1		A0	54	A3	6C		8D	45	BC	AD
8E	5A	A7	5D		FE	2C	39	76		70	76	9E	2B
3E	7A	5A	77		17	B1	39	05		29	CB	63	72

Tabulka 5: Procedura AddRoundKey

# O čem to bude



- 1 Potřeba a historie vzniku AES
- 2 Popis šifrovacího algoritmu AES
- 3 Tvorba klíčů

## 4 Šifrovací proces

## 5 Dešifrovací proces

- Procedura InvAddRoundKey
- Procedura InvMixColumns
- Procedura InvShiftRows
- Procedura InvSubBytes

# Procedura InvAddRoundKey

Dešifrovací proces probíhá inverzně k šifrovacímu. Procedury po sobě následují v opačném pořadí, InvMixColumns se vynechává v první rundě.

# Procedura InvAddRoundKey

Dešifrovací proces probíhá inverzně k šifrovacímu. Procedury po sobě následují v opačném pořadí, InvMixColumns se vynechává v první rundě.

Jednotlivé prvky pole State se přičtou pomocí operace Xor ke stejným prvkům příslušného rundovního klíče.

# Procedura InvAddRoundKey

Dešifrovací proces probíhá inverzně k šifrovacímu. Procedury po sobě následují v opačném pořadí, InvMixColumns se vynechává v první rundě.

Jednotlivé prvky pole State se přičtou pomocí operace Xor ke stejným prvkům příslušného rundovního klíče.

6B	F1	49	BA	$\oplus$	A0	88	23	2A	=	CB	79	6A	90
8D	45	BC	AD		A0	54	A3	6C		77	11	1F	C1
70	76	9E	2B		FE	2C	39	76		8E	5A	A7	5D
29	CB	63	72		17	B1	39	05		3E	7A	5A	77

Tabulka 6: Procedura InvAddRoundKey

# Procedura InvMixColumns

V proceduře InvMixColumns dojde ke změně jednotlivých sloupců pole State. Každý bajt se změní na novou hodnotu, která je funkcí všech čtyř bajtů sloupce.



# Procedura InvMixColumns

V proceduře InvMixColumns dojde ke změně jednotlivých sloupců pole State. Každý bajt se změní na novou hodnotu, která je funkcí všech čtyř bajtů sloupce.

Stejně jako v proceduře MixColumns se i zde operace provádí modulo  $m(x) = x^8 + x^4 + x^3 + x^1 + 1$ .

# Procedura InvMixColumns

V proceduře InvMixColumns dojde ke změně jednotlivých sloupců pole State. Každý bajt se změní na novou hodnotu, která je funkcí všech čtyř bajtů sloupce.

Stejně jako v proceduře MixColumns se i zde operace provádí modulo  $m(x) = x^8 + x^4 + x^3 + x^1 + 1$ .

CB	79	6A	90
77	11	1F	C1
8E	5A	A7	5D
3E	7A	5A	77

$$\Rightarrow \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \cdot \begin{bmatrix} CB \\ 77 \\ 8E \\ 3E \end{bmatrix} = \begin{bmatrix} CB \\ CD \\ 96 \\ 9C \end{bmatrix}$$

Tabulka 7: Procedura InvMixColumns

# Procedura InvShiftRows

Při proceduře InvShiftRows dojde k posunu v rámci řádků v poli State.

# Procedura InvShiftRows

Při proceduře InvShiftRows dojde k posunu v rámci řádků v poli State.

První řádek zůstává stejný, druhý řádek se posune o jednu pozici doprava, třetí řádek o dvě pozice a čtvrtý řádek o tři pozice.

# Procedura InvShiftRows

Při proceduře InvShiftRows dojde k posunu v rámci řádků v poli State.

První řádek zůstává stejný, druhý řádek se posune o jednu pozici doprava, třetí řádek o dvě pozice a čtvrtý řádek o tři pozice.

CB	1B	3D	A5		CB	1B	3D	A5
CD	0E	DF	82	⇒	82	CD	0E	DF
96	A8	FB	9E	⇒ ⇒	FB	9E	96	A8
9C	F5	91	C2	⇒ ⇒ ⇒	F5	91	C2	9C

Tabulka 8: Procedura InvShiftRows

# InvSubBytes I

Pro proceduru InvSubBytes byla vytvořena inverzní substituční tabulka InvS-Box.

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

# InvSubBytes II

Všechny prvky pole State nahradíme pomocí InvS-Boxu. První čtyři bity prvku označují řádek v tabulce InvS-Box, další čtyři sloupec.

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

# InvSubBytes III

Například prvek "D4"ukazuje v S-Boxu na čtrnáctý řádek, čtvrtý sloupek a to je hodnota "19".

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



# InvSubBytes IV

CB	1B	3D	A5	⇒	59	44	8B	29
82	CD	0E	DF		11	80	D7	EF
FB	9E	96	A8		63	DF	35	6F
F5	91	C2	9C		77	AC	A8	1C

Tabulka 9: Procedura InvSubBytes

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73