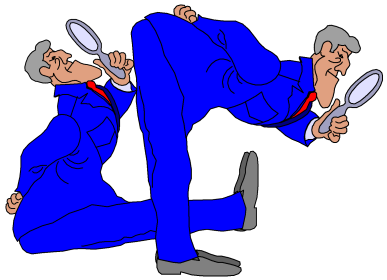
10. Šifrovací algoritmus AES

Jan Paseka

Ústav matematiky a statistiky
Masarykova univerzita

7. prosince 2023

O čem to bude



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 Rinjdael. Název vznikl přesmyčkou jmen jejích dvou belgických autorů Vincenta Rijnmana 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 Rinjdael. Název vznikl přesmyčkou jmen jejích dvou belgických autorů Vincenta Rijnmana 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

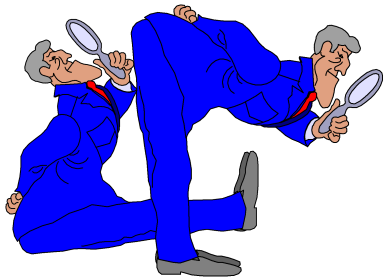
Po pěti letech byla šifra AES schválena jako nejvhodnější z patnácti návrhů. Dne 26. května 2002 začala být ke svému účelu používána jako federální standard USA. AES je první šifra, dostupná široké veřejnosti, která je zároveň uznána Národní bezpečností agenturou NSA.

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

O čem to bude



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 4×4 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 4×4 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 4×4 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 α splňuje rovnici $\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1 = 0.$

Bloková šifra III

Speciálně jeden byte $(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$ odpovídá prvku

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

Bloková šifra III

Speciálně jeden byte $(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$ odpovídá prvku

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

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

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

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

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

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.

Bloková šifra IV

Algorithm 1: Šifrování pomocí AES

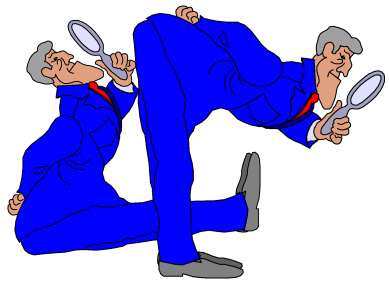
input : Klíč K, Zpráva M

output: Kryptogram C

1 $(K_1, \ldots, K_{10}) \leftarrow \mathit{Expand}(K)$
2 $s \leftarrow M \oplus K_0$
3 **for** $r = 1$ **to** *10* **do**
4 $\quad s \leftarrow \mathit{SubBytes}(s)$
5 $\quad s \leftarrow \mathit{ShiftRows}(s)$
6 $\quad$ **if** $r \leq 9$ **then** $s \leftarrow \mathit{MixColumns}(s)$
7 $\quad s \leftarrow s \oplus K_r$
8 **return** $C \leftarrow s$

See ▸ an animation of Rijndael!

O čem to bude



- Tvorba nových klíčů

Tvorba nových klíčů I

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

Tvorba nových klíčů I

Šifrovací klíč zadáváme na začátku, před započetím 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 započetím 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 započetím 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")

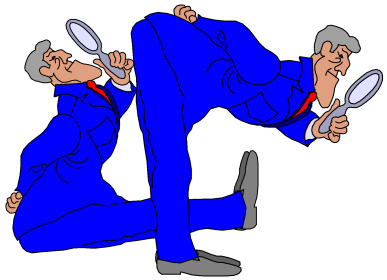
Tvorba nových klíčů III

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

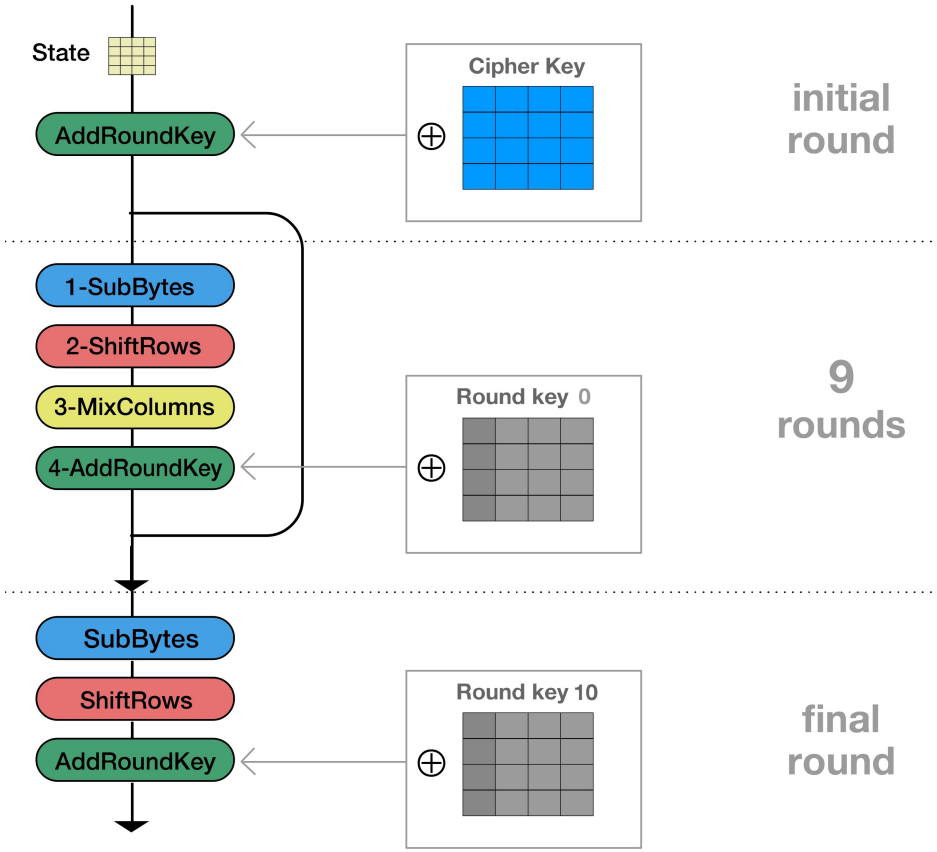
O čem to bude



- Procedura SubBytes
- Procedura ShiftRows
- Procedura MixColumns
- Procedura AddRoundKey

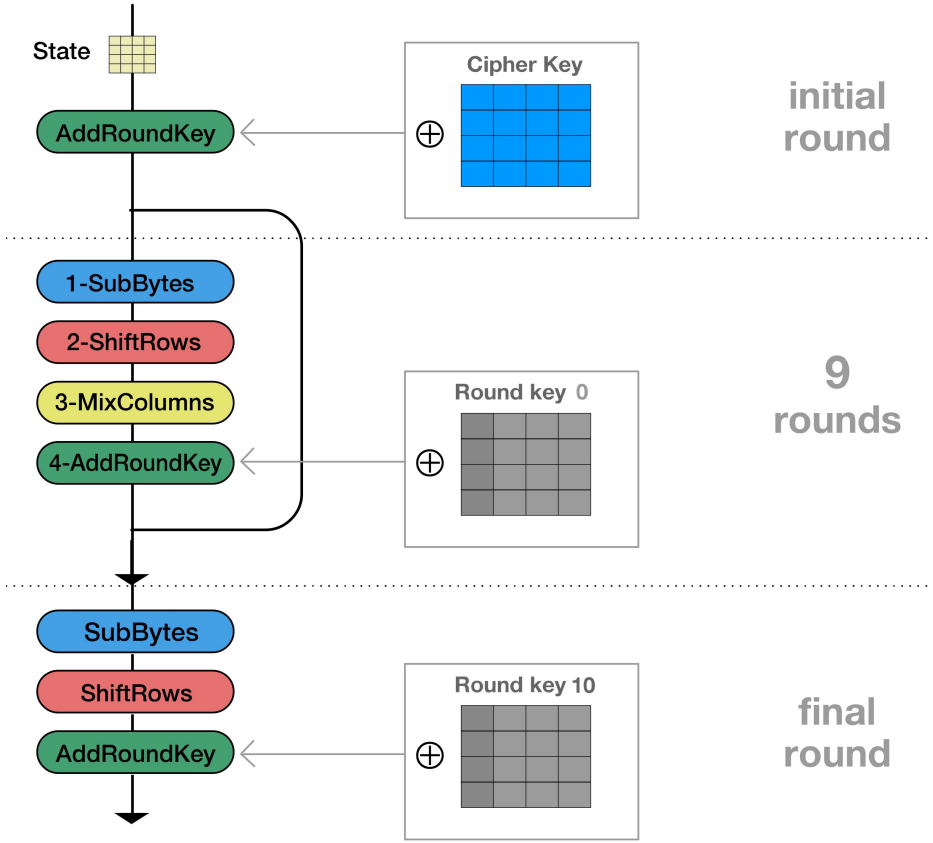
Čtyři procedury I

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



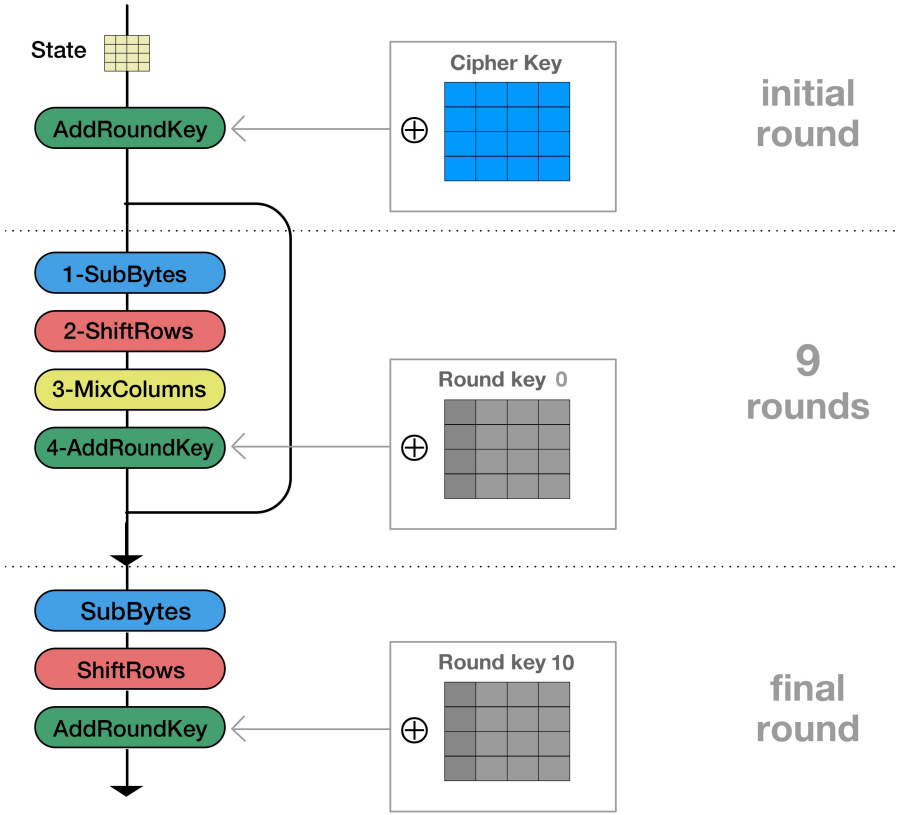
Čtyři procedury II

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



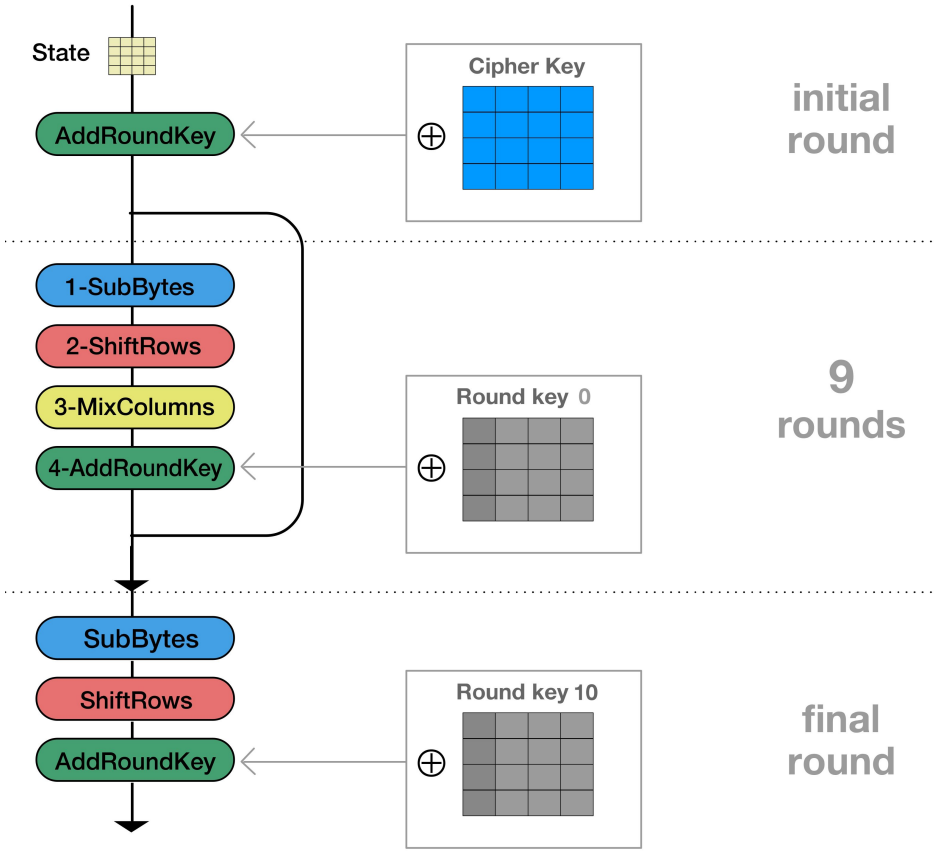
Čtyři procedury III

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



Čtyři procedury IV

Vyjímku 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 $\overline{b}$ předpisem

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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
	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
x	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

Má 16 řádků a 16 sloupců.

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
x	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
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

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															
x	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

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															
x	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

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															
x	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

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

$\Rightarrow$

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

Tabulka 2: Procedura SubBytes

hex	y															
x	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

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	$\Leftarrow$			CD	0E	DF	82
FB	9E	96	A8	$\Leftarrow$	$\Leftarrow$		96	A8	FB	9E
F5	91	C2	9C	$\Leftarrow$	$\Leftarrow$	$\Leftarrow$	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 $\overline{s}_j$ tak, že

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

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

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

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 přislušného rundovního klíče.

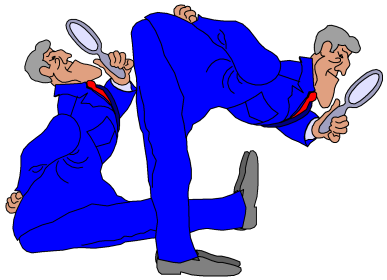
Procedura AddRoundKey I

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

CB	79	6A	90	⊕	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



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 přislušného 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 přislušného rundovního klíče.

6B	F1	49	BA		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
CD	0E	DF	82
96	A8	FB	9E
9C	F5	91	C2

$\Rightarrow \quad \Rightarrow \quad \Rightarrow \quad \Rightarrow$

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

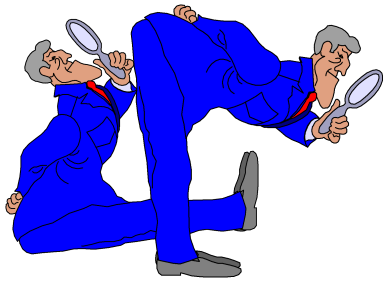
$\Rightarrow$

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

Tabulka 9: Procedura InvSubBytes

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

O čem to bude



Diferenciální a lineární kryptoanalýza

Odkaz: “A Tutorial on Linear and Differential Cryptanalysis” `www.engr.mun.ca/~howard/PAPERS/ldc_tutorial.pdf`

- Diferenciální a lineární kryptoanalýza jsou techniky pro útok na blokové šifry. Diferenciální kryptoanalýza byla poprvé představena Elim Bihamem a Adi Shamirem v roce 1991. Diferenciální kryptoanalýza funguje tak, že hledá páry prostých textů, které mají určitý rozdíl ve svých vstupních hodnotách, ale odpovídající šifrovací texty mají určitý rozdíl ve svých výstupních hodnotách. Tyto páry prostých textů a šifrovacích textů se nazývají diferenciály.
- Cílem diferenciální kryptoanalýzy je najít velký počet diferenciálů, které mají vysokou pravděpodobnost výskytu. Pokud je možné najít dostatečně velký počet diferenciálů, pak je útočník může použít k útoku na šifru.

Lineární kryptoanalýza I

- Lineární kryptoanalýza je další technikou pro útok na blokové šifry. Poprvé ji představil Mitsuru Matsui v roce 1993. Lineární kryptoanalýza funguje hledáním lineárních aproximací šifry. Lineární aproximace je rovnice, která spojuje vstupní, výstupní a klíčové bity šifry.
- Cílem lineární kryptoanalýzy je najít lineární aproximaci, která má vysokou korelaci s šifrou. Pokud lze najít lineární aproximaci s vysokou korelaci, pak ji útočník může použít k útoku na šifru.
- Lineární kryptoanalýza je výkonná technika, která byla použita k prolomení řady blokových šifer. Je však důležité si uvědomit, že lineární kryptoanalýza není všemocná a nefunguje na všechny šifry.

Lineární kryptoanalýza II

- Lineární vztahy mezi vstupem X a výstupem Y: bitové pozice $i_1, ..., i_\ell$ a $i'_1, ..., i'_\ell$ mají **bias** p pokud pro náhodně zvolený vstup X a klíč K platí $Y = F_K(X)$,

$$\Pr[X_{i_1} \oplus \cdots \oplus X_{i_\ell} \oplus Y_{i'_1} \oplus \cdots \oplus Y_{i'_\ell} = 0] = p + \frac{1}{2}.$$

- Nevyžaduje se chosen-plaintext attack, known-plaintext attack je dostačující.
- Kroky útoku:
  1. Vytvořte tabulku lineárních aproximací S-boxů.
  2. Vytvořte lineární aproximaci prvních $r - 1$ kol s velkým biasem.
  3. Extrahujte bity dílčího klíče posledního kola, které dobře vyhovují lineární aproximaci.

Příklad lineární analýzy *S*-boxu

X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4	$X_2 \oplus X_3$	$Y_1 \oplus Y_3 \oplus Y_4$	$X_1 \oplus X_4$	Y_2	$X_3 \oplus X_4$	$Y_1 \oplus Y_4$
0	0	0	0	1	1	1	0	0	0	0	1	0	1
0	0	0	1	0	1	0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1	1	0	0	1	1	0
0	0	1	1	0	0	0	1	1	1	1	0	0	1
0	1	0	0	0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	1	1	1	1	1	1	1	1	0
0	1	1	0	1	0	1	1	0	1	0	0	1	0
0	1	1	1	1	0	0	0	0	1	1	0	0	1
1	0	0	0	0	0	1	1	0	0	1	0	0	1
1	0	0	1	1	0	1	0	0	0	0	0	1	1
1	0	1	0	0	1	1	0	1	1	1	1	1	0
1	0	1	1	1	1	0	0	1	1	0	1	0	1
1	1	0	0	0	1	0	1	1	1	1	1	0	1
1	1	0	1	1	0	0	1	1	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1	0	1	0
1	1	1	1	0	1	1	1	0	0	0	1	0	1

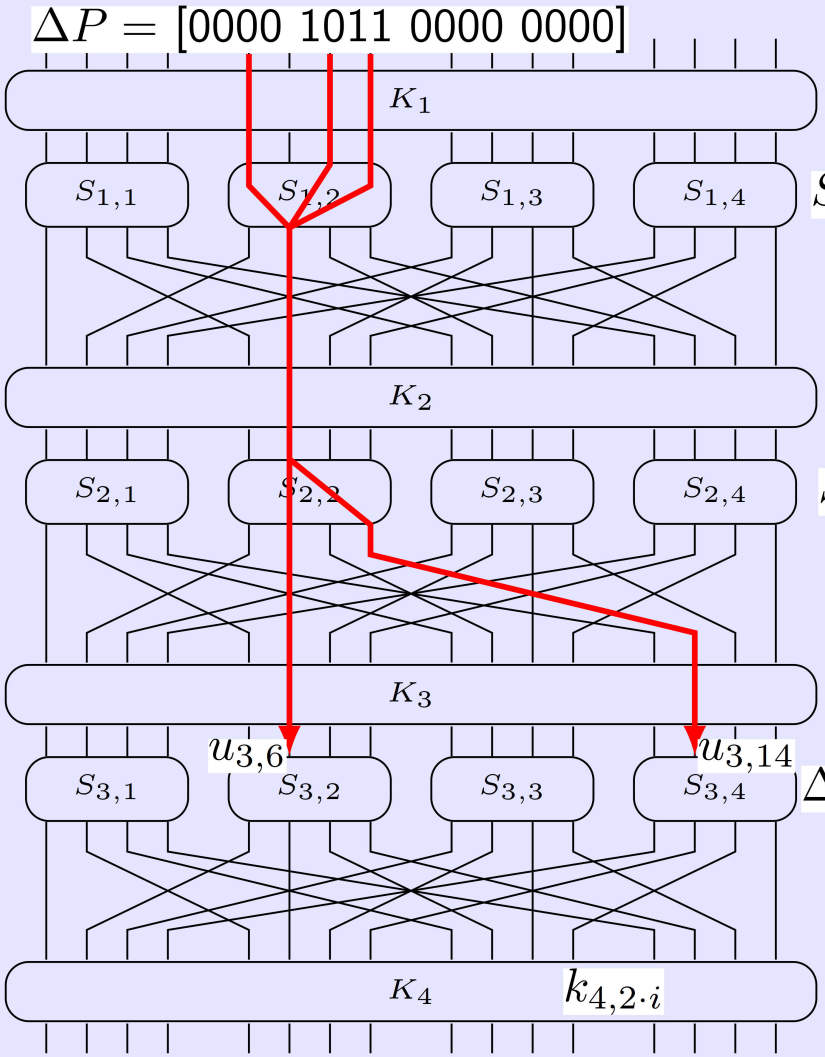
Příklad tabulky lineární distribuce

Input Sum \ Output Sum	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	+8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	−2	−2	0	0	−2	+6	+2	+2	0	0	+2	+2	0	0
2	0	0	−2	−2	0	0	−2	−2	0	0	+2	+2	0	0	−6	+2
3	0	0	0	0	0	0	0	0	+2	−6	−2	−2	+2	+2	−2	−2
4	0	+2	0	−2	−2	−4	−2	0	0	−2	0	+2	+2	−4	+2	0
5	0	−2	−2	0	−2	0	+4	+2	−2	0	−4	+2	0	−2	−2	0
6	0	+2	−2	+4	+2	0	0	+2	0	−2	+2	+4	−2	0	0	−2
7	0	−2	0	+2	+2	−4	+2	0	−2	0	+2	0	+4	+2	0	+2
8	0	0	0	0	0	0	0	0	−2	+2	+2	−2	+2	−2	−2	−6
9	0	0	−2	−2	0	0	−2	−2	−4	0	−2	+2	0	+4	+2	−2
A	0	+4	−2	+2	−4	0	+2	−2	+2	+2	0	0	+2	+2	0	0
B	0	+4	0	−4	+4	0	+4	0	0	0	0	0	0	0	0	0
C	0	−2	+4	−2	−2	0	+2	0	+2	0	+2	+4	0	+2	0	−2
D	0	+2	+2	0	−2	+4	0	+2	−4	−2	+2	0	+2	0	0	+2
E	0	+2	+2	0	−2	−4	0	+2	−2	0	0	−2	−4	+2	−2	0
F	0	−2	−4	−2	−2	0	+2	0	0	−2	+4	−2	−2	0	+2	0

$x_2 \oplus x_3 = y_1 \oplus y_3 \oplus y_4$ - 12 $\times$, bias bude $12 - 8 = 4$ $\times$,

$x_2 \oplus x_3 : 0110 = 6, \quad y_1 \oplus y_3 \oplus y_4 : 1011 = B$, tj. $(6, B) = 4$

Příklad lineární kryptoanalýzy



$S_{1,2}$: $\Delta X = B \rightarrow \Delta Y = 4$

$S_{2,2}$: $\Delta X = 4 \rightarrow \Delta Y = 5$

$\Delta U = [0000\ 0100\ 0000\ 0100]$

Guess $k_{4,2\cdot i}$

Diferenciální kryptoanalýza I

- Konkrétní rozdíly Δ_x na vstupu, které vedou ke konkrétním rozdílům Δ_y na výstupu s pravděpodobností $p \gg 2^{-n}$.
- $x_1 \oplus x_2 = \Delta_x$, $F_k(x_1) \oplus F_k(x_2) = \Delta_y$ s pravděpodobností p.
- Tohoto lze využít pomocí CPA útoku.
- Kroky útoku:
  1. Vytvořte tabulku distribuce rozdílů S-boxů.
  2. Vytvořte diferenciální charakteristiku prvních $r - 1$ kol s velkým biasem.
  3. Extrakce bitů podklíče posledního kola, které dobře vyhovují diferenciálním charakteristikám.

Příklad diferenciální analýzy *S*-boxu

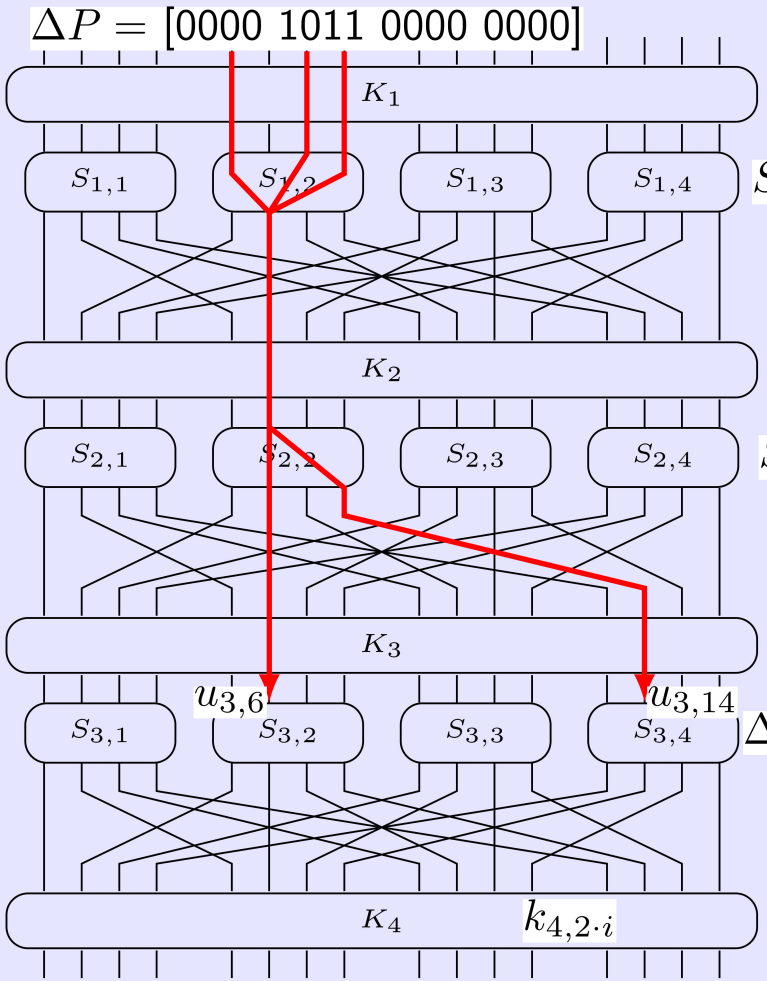
X	Y	ΔY		
		$\Delta X = 1011$	$\Delta X = 1000$	$\Delta X = 0100$
0000	1110	0010	1101	1100
0001	0100	0010	1110	1011
0010	1101	0111	0101	0110
0011	0001	0010	1011	1001
0100	0010	0101	0111	1100
0101	1111	1111	0110	1011
0110	1011	0010	1011	0110
0111	1000	1101	1111	1001
1000	0011	0010	1101	0110
1001	1010	0111	1110	0011
1010	0110	0010	0101	0110
1011	1100	0010	1011	1011
1100	0101	1101	0111	0110
1101	1001	0010	0110	0011
1110	0000	1111	1011	0110
1111	0111	0101	1111	1011

Příklad tabulky distribuce rozdílů

Input Difference	Output Difference 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

$\Delta X = 1011 = B, \Delta Y = 0010 = 2$ celkem 8 $\times$, tedy $(B, 2) = 8$

Příklad diferenciální kryptoanalýzy



$S_{1,2}$: $\Delta X = B \rightarrow \Delta Y = 4$

$S_{2,2}$: $\Delta X = 4 \rightarrow \Delta Y = 5$

$\Delta U = [0000\ 0100\ 0000\ 0100]$

Guess $k_{4,2\cdot i}$