Šifrovací algoritmy

kódování – způsob zápisu informace pomocí znaků zvolené abecedy – kódu
šifrování – podtřída kódů, k jejichž interpretaci je nutné znát dodatečnou informaci (***klíč***)

Klasifikace šifrovacích algoritmů

- podle způsobu práce – blokové, proudové
- podle klíčů s tajným klíčem (symetrické), s veřejným klíčem (asymetrické)

Kerckhoffův předpoklad:
Útočník zná všechny aspekty šifrovacího algoritmu s výjimkou použitého klíče.

Naivní charakteristika dobré šifry

Množství práce vynaložené na šifrování a dešifrování by mělo být úměrné požadovanému stupni utajení.
Šifrovací algoritmus by neměl obsahovat zbytečná omezení.
Implementace algoritmu by měla být co nejjednodušší.
Chyby při šifrování by se neměly příliš šířit a ovlivňovat následující komunikaci.
Zprávy by se zašifrováním neměly zvětšovat.

Security through obscurity NEFUNGUJE!

Zmatení (confusion) - nelze predikovat, jakou změnu zašifrovaného textu vyvolá byť jen malá změna otevřeného textu ⇔ složitá funkční závislost mezi zašifrovaným textem a párem klíč - otevřený text.

Difuze (diffusion) - změna otevřeného textu se promítá do mnoha míst zašifrovaného textu.

Bezpečný systém - nelze získat otevřený text na základě znalosti odpovídající šifry

kryptoanalytik hledá transformaci $\mathbf{h} : C \rightarrow P$, $\mathbf{h}$ nebývá jednoznačná

Efektivně bezpečný systém - Prob($\mathbf{h}$(C) = P) < ε.

Ideální stav

Perfektní utajení (perfect secrecy) - mějme *n* možných otevřených textů, stejné množství klíčů a možných šifer.

$$Prob_{C_1}\left(\mathbf{h}\left(C_1\right)=P\right)=Prob\left(\mathbf{h}\left(C_1\right)=P\right)=Prob\left(P\right)$$

Redundance

počet bitů nutný k reprezentaci všech znaků abecedy $A=\left\lceil \mathbf{log}_2\left(k\right)\right\rceil$

počet všech možných zpráv délky $n = 2^{An}$, z toho 2^{Rn} smysluplných. *R* nazýváme *rate* jazyka. Redundance je definována

$$D = A - R$$

Pokud algoritmus šifruje několik různých zpráv, z nichž jedna je smysluplná, do stejné šifry, systém může být bezpečný.

Formálnější charakteristika dobré šifry

Kryptosystém je trojice (G, E, D) pravděpodobnostních p-time algoritmů splňující následující kritéria:

- Algoritmus G (generátor klíčů) nad vstupen 1^n vytvoří dvojici bitových řetězců
- Pro každý pár (*e, d*) z oboru hodnot G(1^n) a pro každé $\alpha \in \{1, 0\}^*$ algoritmus E (šifrování) a D (dešifrování) splňují
  $\Pr(D(d, E(e, \alpha)) = \alpha) = 1$
  kde pravděpodobnost se bere přes interní náhodná rozhodnutí algoritmů E a D.

Kryptosystém (G, E, D) je ***sémanticky bezpečný*** pokud existuje p-time transformace *T* taková že pro každý polynomiálně velký obvod $\{C_n\}$, každou posloupnost $\{X_n\}_{n\in N}$ kde $|X_n|$ je polynomiálně omezeno, každý pár polynomiálních funkcí *f* a *h:* $\{1, 0\}^* \rightarrow \{1, 0\}^*$, každý polynom *p* a všechna dostatečně velká *n*

$$\Pr\left(C_n\left(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)\right)=f(X_n)\right) < \Pr\left(C'_n\left(1^{|X_n|}, h(X_n)\right)=f(X_n)\right)+\frac{1}{p(n)},$$

kde $C'_n = T(C_n)$ je obvod vytvořený transformací *T* nad vstupen C_n. Funkce *h* poskytuje parciální informaci o plaintextu X_n.

Kryptosystém (G, E, D) poskytuje ***nerozlišitelné šifrování***, pokud pro každý polynomiálně velký obvod $\{C_n\}$, každý polynom p, všechna dostatečné velká n, každé $x, y \in \{0,1\}^{poly(n)}$ (tj. "stejně dlouhá") a $z \in \{0,1\}^{poly(n)}$

$$\left|\Pr\left(C_n\left(z, E_{G_1(1^n)}(x)\right)=1\right)-\Pr\left(C_n\left(z, E_{G_1(1^n)}(y)\right)=1\right)\right|<\frac{1}{p(n)}$$

Pravděpodobnost se bere přes náhodná rozhodnutí algoritmů G a E.
Lze ukázat, že obě definice jsou ekvivalentní.

Šifrování obecně:

- *neskrývá samu existenci informace*
- *nezaručuje integritu*
- *nezaručuje autenticitu (původ)*
- *nebrání proti fabrikaci*
- *neskrývá všechny vlastnosti plaintextu*
- *důvěrnost zachovává pouze za určitých podmínek*
- *... a zbyde vám klíč*

Blokové kryptosystémy s tajným klíčem

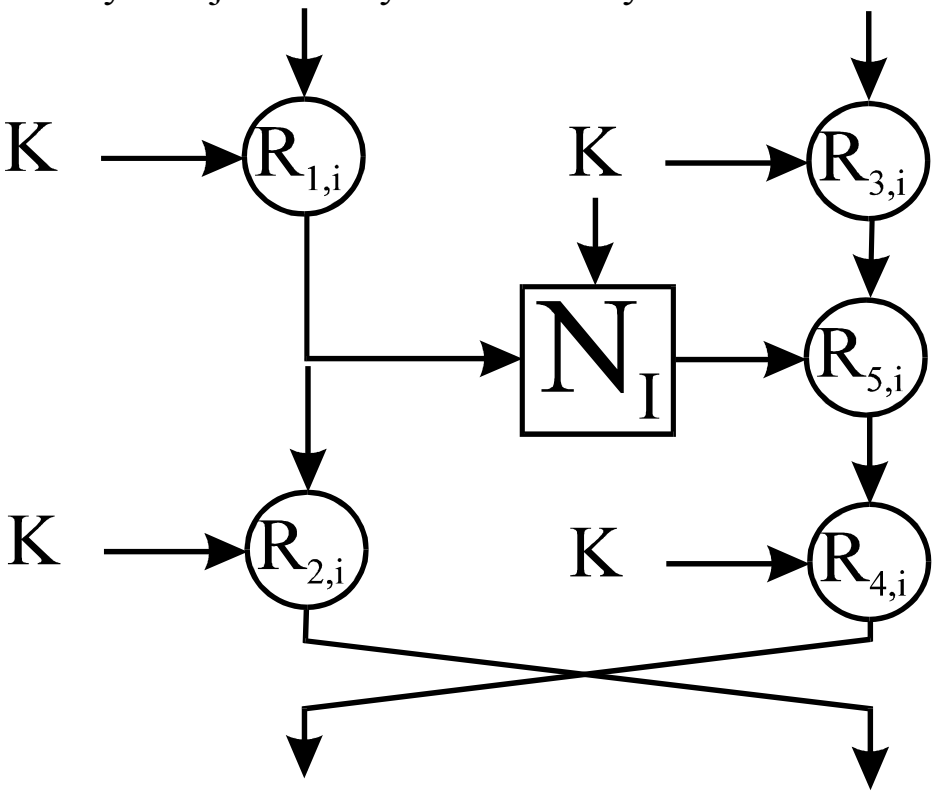
stejný klíč použit pro šifrování a dešifrování
mapují *n*-bitový plaintext na *n*-bitovou šifru za použití *k*-bitového klíče (parametr)
lze chápat jako jednoduché substituční šifry s nad obrovskou abecedou
každá šifra je soustavou bijekcí definujících permutaci nad *n*-bitovými vektory, tj. je invertibilní, klíč vybírá konkrétní bijekci

Hodnocení blokových šifer

- odhadovaná úroveň bezpečnosti ... důvěra v historickou bezpečnost roste s časem
- velikost klíče – je horním limitem bezpečnosti šifry
- výkon (efektivita) – měřeno počtem instrukcí na zašifrovaný byte
- velikost bloku
- komplexita kryptografické transformace
- zvětšení dat šifrováním
- propagace chyb
- komplexita expanze klíče (inicializace)

mnoho dnešních systémů jsou *Feistelovy šifry*

obecný tvar jednoho cyklu Feistelovy sítě:



$R_{j,i}$ - reversibilní funkce textu a klíče
N_i - nereversibilní funkce textu a klíče

Kryptosystém DES

Vyvinula firma IBM na zakázku NBS počátkem 70. let. Původní název DEA, v USA DEA1. Jako standard přijat 23. 11. 1976
Patrně nejrozsáhleji používaný šifrovací algoritmus všech dob.
Přibližně od přelomu tisíciletí **není považován za bezpečný z důvodu malé délky klíče** – nicméně 20 let intenzivní kryptoanalýzy nepřineslo žádný zásadní poznatek o vnitřních slabinách algoritmu
Norma ANSI X3.92

šifruje 64-bitové bloky otevřeného textu na 64-bitové výstupní bloky, délka klíče 64 bitů

Požadavky zadavatele

1. Alg. musí poskytovat vysoký stupeň ochrany
2. Musí být formálně popsatelný a snadno pochopitelný
3. Bezpečnost algoritmu nesmí záviset na znalosti či neznalosti samotného algoritmu.
4. Musí být dostupný pro nejširší veřejnost.

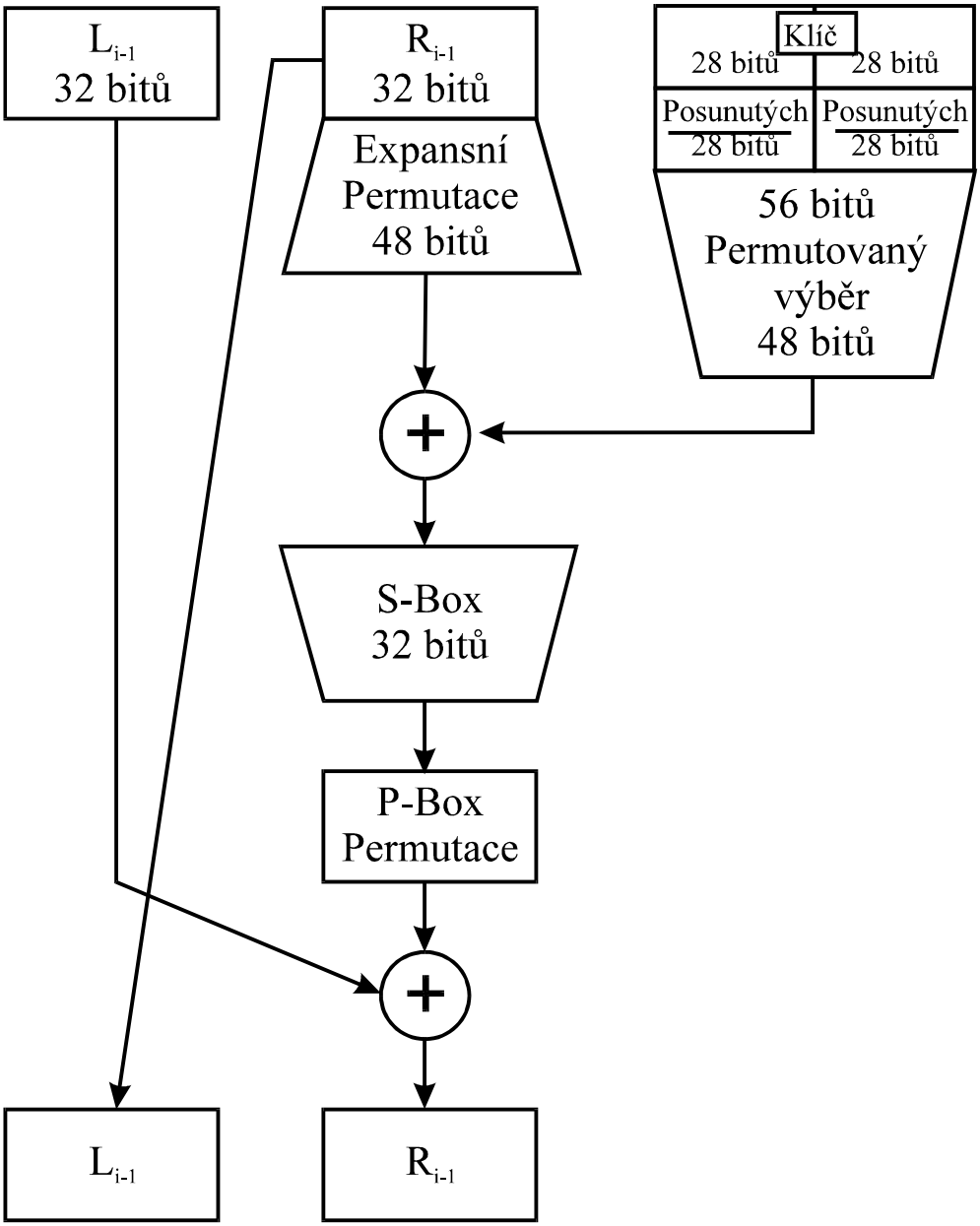
5. Musí být použitelný v nejrůznějších aplikacích.
6. Musí být efektivní.
7. Musí být ověřitelný.
8. Musí být exportovatelný.

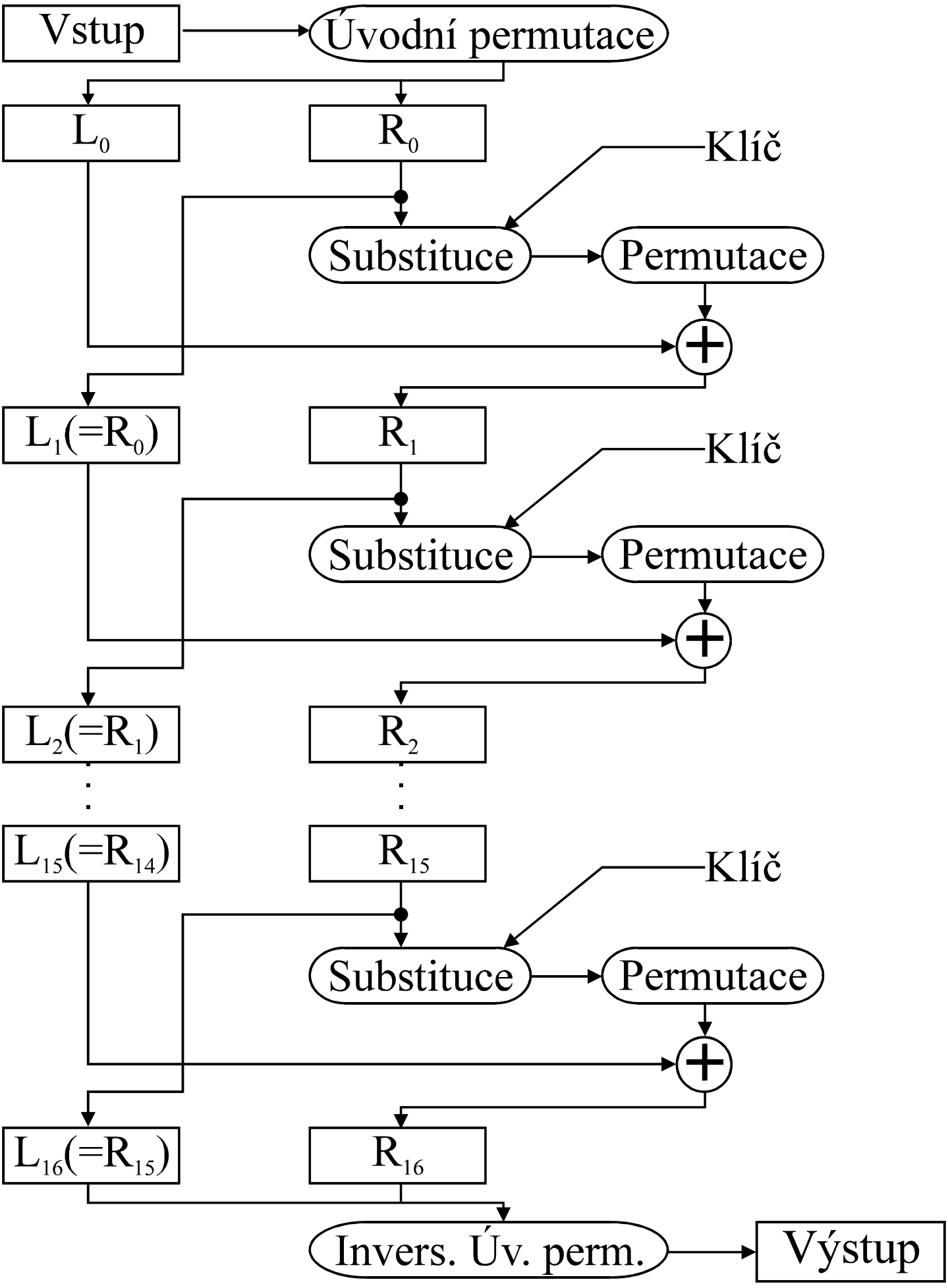
Analýza

- úvodní permutace nemá prakticky žádný vliv
- příliš krátký klíč, navíc efektivně pouze 56-bitový
- komplementárnost - $C = \mathbf{E}(K, P) \Leftrightarrow \neg C = \neg\, \mathbf{E}(\neg K, \neg P)$
- existence slabých (weak) klíčů ($\mathbf{E}(K) = \mathbf{D}(K)$) a poloslabých (semiweak) klíčů ($\mathbf{E}(K_1)\ \mathbf{E}(K_2)) = Id$.
- nevhodný návrh S-boxů

Možné způsoby zvýšení bezpečnosti

1. vícenásobné šifrování - nestačí dvojnásobné, ke skutečnému zvýšení bezpečnosti nutno šifrovat
$$C = \mathbf{E}(K_1)\mathbf{D}(K_2)\mathbf{E}(K_1)$$
2. zvětšení délky hesla na 768 bitů - nepříliš účinné


L_{i-1}
32 bitů
R_{i-1}
32 bitů
Klíč
28 bitů
28 bitů
Posunutých
28 bitů
Posunutých
28 bitů
Expansní
Permutace
48 bitů
56 bitů
Permutovaný
výběr
48 bitů
+
S-Box
32 bitů
P-Box
Permutace
+
L_{i-1}
R_{i-1}


Vstup
Úvodní permutace
L0
R0
Klíč
Substituce
Permutace
L1(=R0)
R1
Klíč
Substituce
Permutace
L2(=R1)
R2
L15(=R14)
R15
Klíč
Substituce
Permutace
L16(=R15)
R16
Invers. Úv. perm.
Výstup

Poučení z DESu

- paralelní aplikace malých S-Boxů je nevhodná pro SW implementaci
- příliš malý klíč
- permutace a permutační výběry těžko zvládnutelné v SW
- Feistelova síť principiálně funguje a vykazuje dobrou odolnost vůči analýze
- nové algoritmy by měly být vhodné pro SW implementaci na běžném HW (výběr operací)
- je třeba zvýšit odolnost vůči masivně paralelnímu útoku (drahá key schedule)
- rozhodně delší klíče, případně blok
- vysokou odolnost vykazují širší S-boxy a na klíči závislé interní struktury algoritmu
- nahrazení paralelních operací, které jsou neefektivní a mají omezený lavinový efekt sekvenčními operacemi

Systém Blowfish

opět Feistelova šifra, délka bloku 64 bitů, proměnná délka klíče až 448 bitů

Subklíče

předpočítávají se před každým šifrováním

- *P-pole* = 32 bitové klíče
  P_1 , P_2 , … P_{18}

- pole *S-boxů*, každý 256 32-bitových položek
  $S_{1,0}$, $S_{1,1}$, … $S_{1,255}$
  $S_{2,0}$, $S_{2,1}$, … $S_{2,255}$
  $S_{3,0}$, $S_{3,1}$, … $S_{3,255}$
  $S_{4,0}$, $S_{4,1}$, … $S_{4,255}$

Nereverzibilní funkce F:

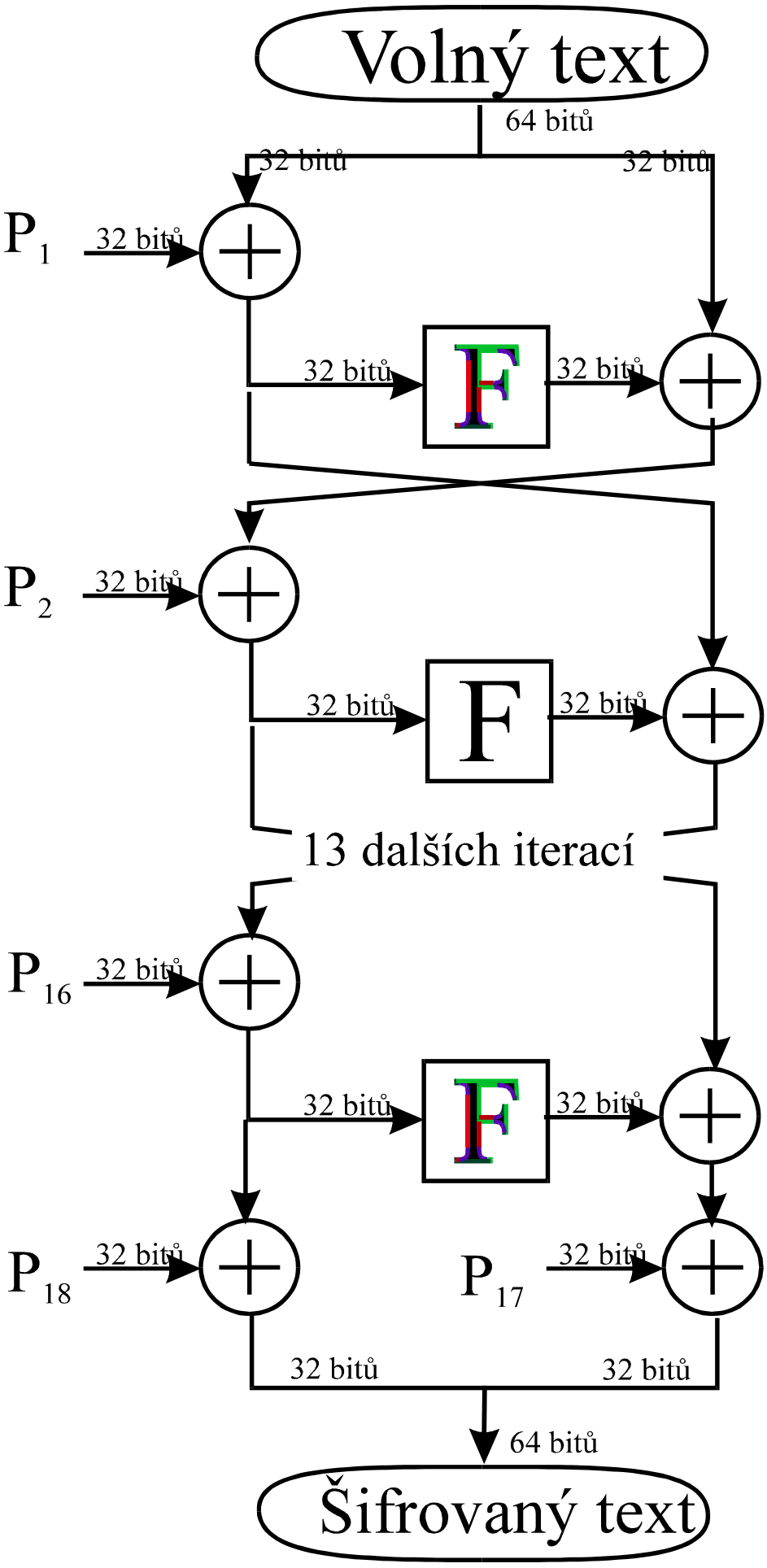
$$\mathrm{F}(x_L) = \left(\left(S_{1,a} + S_{2,b} \bmod 2^{32}\right) \mathrm{xor}\, S_{3,c}\right) + S_{4,d} \bmod 2^{32}$$

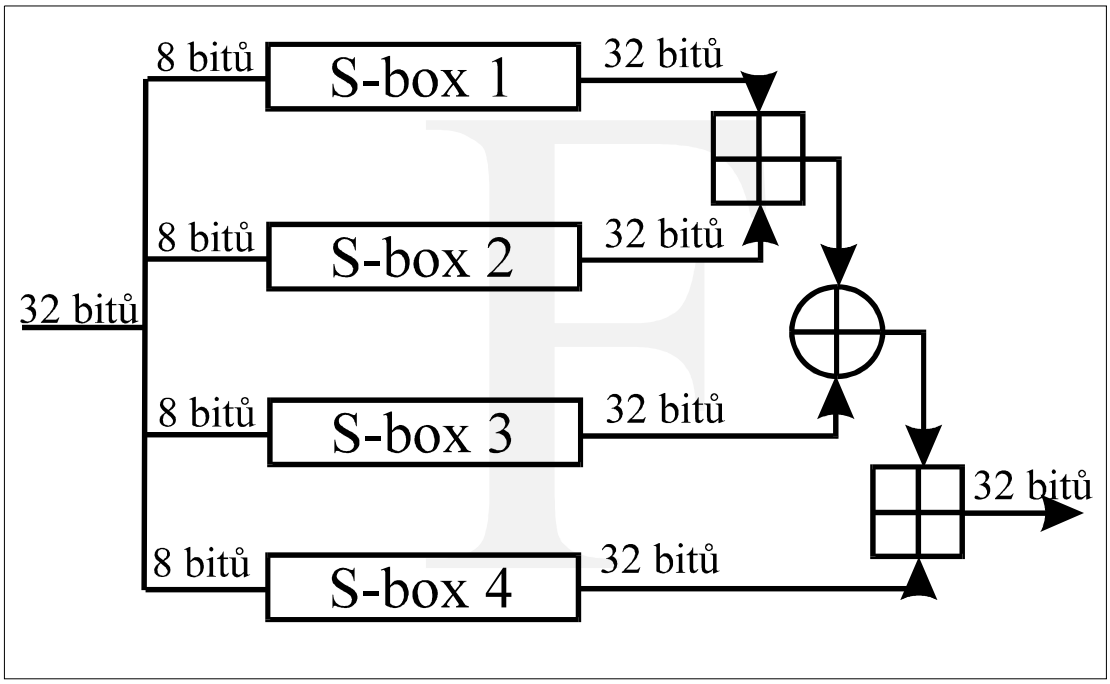
$$x_L = a|b|c|d$$

Generování podklíčů

1. Inicializujeme *P-pole* a všechny *S-boxy* pevným řetězcem
2. **xor** P_1 s prvními 32 bity klíče, **xor** P_2 s dalšími 32 bity klíče atd.
3. Zašifrujeme nulový řetězec
4. P_1 a P_2 nahradíme výstupem předchozího kroku
5. zašifrujeme výstup kroku 3.
6. P_3 a P_4 nahradíme výstupem předchozího kroku
7. stejně pro ostatní položky *P-pole* a všechny *S-boxy*

Algoritmus provádí 16 cyklů nad vstupem délky 64 bitů. Pro účely analýzy navrženy jeho zmenšené varianty (MiniBlowfish) pracující nad vstupem 32 popřípadě 16 bitů.

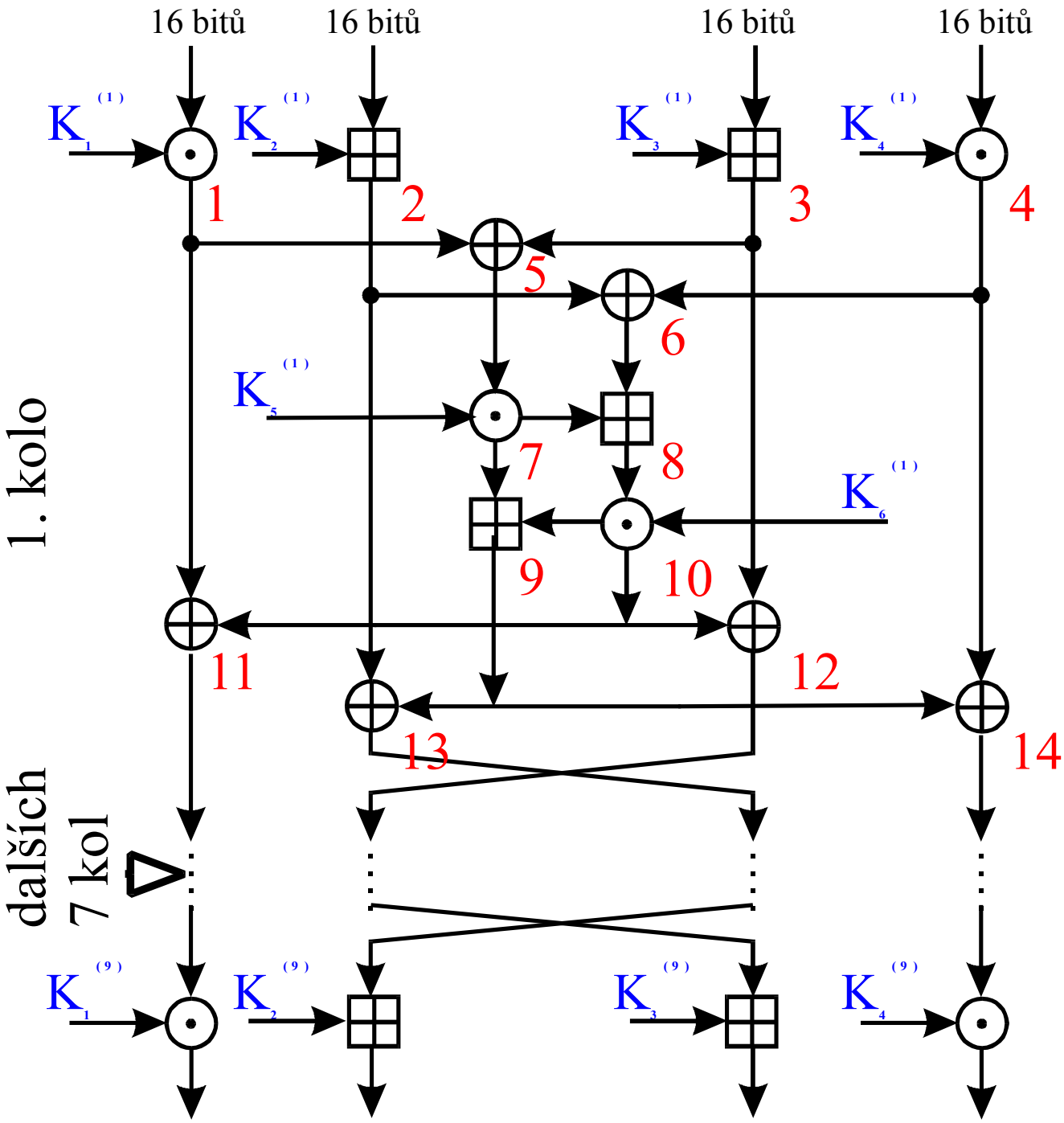

Volný text
64 bitů
32 bitů
32 bitů
P1
32 bitů
32 bitů
F
32 bitů
P2
32 bitů
32 bitů
F
32 bitů
13 dalších iterací
P16
32 bitů
32 bitů
F
32 bitů
P18
32 bitů
P17
32 bitů
32 bitů
32 bitů
64 bitů
Šifrovaný text



Kryptosystém IDEA

publikován v roce 1991 pod názvem IPES, autoři X. Lai a J. Massey
současný název akronymem za International Data Encryption Algorithm
bloková šifra s délkou bloku 64 bitů, pracující s klíčem o délce 128 bitů
algoritmus byl patentován (patentní ochrana již vypršela), nešel volně používat

Blok otevřeného textu je rozdělen na čtyři části, každá o délce šestnáct bitů. Poté je provedeno osm kol šifrovacího procesu.



Tvorba subklíčů

celkem 52 subklíčů:

1. klíč rozdělen na osm částí - vznikne prvních osm subklíčů
2. je provedena rotace klíče vlevo o 25 bitů
3. vzniklý řetězec je opět rozdělen na osm částí – subklíčů
4. Opakováním 2 a 3 získáme potřebné množství subklíčů.

Dešifrování

stejný algoritmus jako pro šifrování, rozdíl pouze v použitých subklíčích

použijeme-li pro šifrování v *i*-tém ($i = 1, \ldots, 8$) kole klíčů

$K_1^{(i)}, K_2^{(i)}, K_3^{(i)}, K_4^{(i)}, K_5^{(i)}, K_6^{(i)}$

a v závěrečné fázi $K_1^{(9)}, K_2^{(9)}, K_3^{(9)}, K_4^{(9)}$,

pro dešifrování použijeme klíčů ve tvaru

round	$\bar{K}_1^{(r)}$	$\bar{K}_2^{(r)}$	$\bar{K}_3^{(r)}$	$\bar{K}_4^{(r)}$	$\bar{K}_5^{(r)}$	$\bar{K}_6^{(r)}$
$r = 1$	$(K_1^{(10-r)})^{-1}$	$-K_2^{(10-r)}$	$-K_3^{(10-r)}$	$(K_4^{(10-r)})^{-1}$	$K_5^{(9-r)}$	$K_5^{(9-r)}$
$2 \leq r \leq 8$	$(K_1^{(10-r)})^{-1}$	$-K_3^{(10-r)}$	$-K_2^{(10-r)}$	$(K_4^{(10-r)})^{-1}$	$K_5^{(9-r)}$	$K_5^{(9-r)}$
$r = 9$	$(K_1^{(10-r)})^{-1}$	$-K_2^{(10-r)}$	$-K_3^{(10-r)}$	$(K_4^{(10-r)})^{-1}$	—	—

kde $(K_x^{(i)})^{-1}$ znamená multiplikativní inverzi mod $2^{16}+1$, $-K_x^{(i)}$ aditivní inverzi mod 2^{16}

IDEA může být používána v libovolném pracovním módu pro blokové šifry, zejména v módech ECB, CBC, OFB, CFB
lze použít trojnásobné šifrování EDE Triple-IDEA se dvěma 128-bitovými klíči, použít 52 nezávislých řetězců

Analýza

Je zajímavé, že pokud bychom algoritmus upravili tím způsobem, že zvětšíme délku všech řetězců, se kterými pracuje na dvojnásobek, dojde ke ztrátě bezpečnosti.
Algoritmus je považován za bezpečný. Roku 2007 publikován útok proti algoritmu omezenému na 6 kol, v roce 2012 útok na plnou verzi algoritmu, ovšem s velkou asymptotickou složitostí

RC5

publikoval v roce 1994 R. Rivest, přináší novou myšlenku použití rotací závislých na datech
velmi pružný algoritmus s celou řadou parametrů

- délka šifrovacího klíče (0 až 255 bytů)
- počet kol šifrovacího procesu (opět 0 až 255)
- z hodnot 16, 32, 64, ale i vyšších lze zvolit délku slova, algoritmus zpracovává bloky o délce dvojnásobku slova

⊕ označuje bitové XOR, + značí sčítání modulo délka slova, -- odčítání, *x* <- *y* znamená rotaci řetězce *x* vlevo o *y* bitů a symbol -> rotaci opačným směrem.

Šifrování

Předpokládejme prozatím, že máme k dispozici pole subklíčů S. Nechť se blok otevřeného textu skládá ze dvou částí *A* a *B*. Šifrování probíhá dle následujícího předpisu:

A = *A* + S[0];
B = *B* + S[1];
for *i* = 1 **to** <počet_kol> **do**
    A = ((*A* ⊕ *B*) <- *B*) + S[2*i*];
    B = ((*B* ⊕ *A*) <- *A*) + S[2*i* + 1];

Dešifrování

for *i* = <počet_kol> **downto** 1 **do**
    B = ((*B* -- S[2*i* + 1]) -> *A*) ⊕ *A*);
    A = ((*A* -- S[2*i*]) -> *B*) ⊕ *B*);
B = *B* -- S[1];
A = *A* -- S[0];

Inicializace

pole subklíčů *S,* pomocné pole *L* o velikosti tolik slov, aby se do něj "vešel" klíč a dvě magická čísla:
P*w* = **Odd**((*e* – 2)2^w) a
Q*w* = **Odd**((*ϕ* – 2)2^w)
kde *e* je základ přirozeného logaritmu (2,718281...), *ϕ* je tzv. zlatý řez (1,618033...) a *w* značí délku slova. Funkce **Odd** vrací nejbližší liché celé číslo.
Do pole L zkopírujeme od začátku šifrovací klíč, a na konci případně doplníme nulami. Pole S naplníme dle předpisu

S[0] = P_w;
for *i* = 1 **to** 2 * (<počet_kol> + 1) -- 1 **do**
    S[*i*] = S[*i* – 1] + Q_w;

a proces generování subklíčů dokončíme promícháním obou polí:

i = *j* = 0;
A = *B* = 0;
for *k* = 1 **to** 3 * **max**(s, l) **do**
    A = S[*i*] = (S[*i*] + *A* + *B*) <- 3;
    B = L[*j*] = (L[*j*] + *A* + *B)* <- (*A* + *B*);
    i = (*i* + 1) **mod**(s);

$$j = (j + 1) \textbf{ mod}(l);$$

kde s resp. l jsou velikosti polí S, resp. L.

Velikost slova je rozumné volit v závislosti na velikosti slova používaného procesoru, 128 bitů pro hašování. Šest kol pro nenáročné aplikace (není bezpečné), 32 pro ty nejnáročnější. Jako rozumná se jeví volba délka slova 32 bitů, 12 kol, 16bajtový klíč, což krátce zapíšeme RC5-32/12/16.

Poučení z devadesátých let

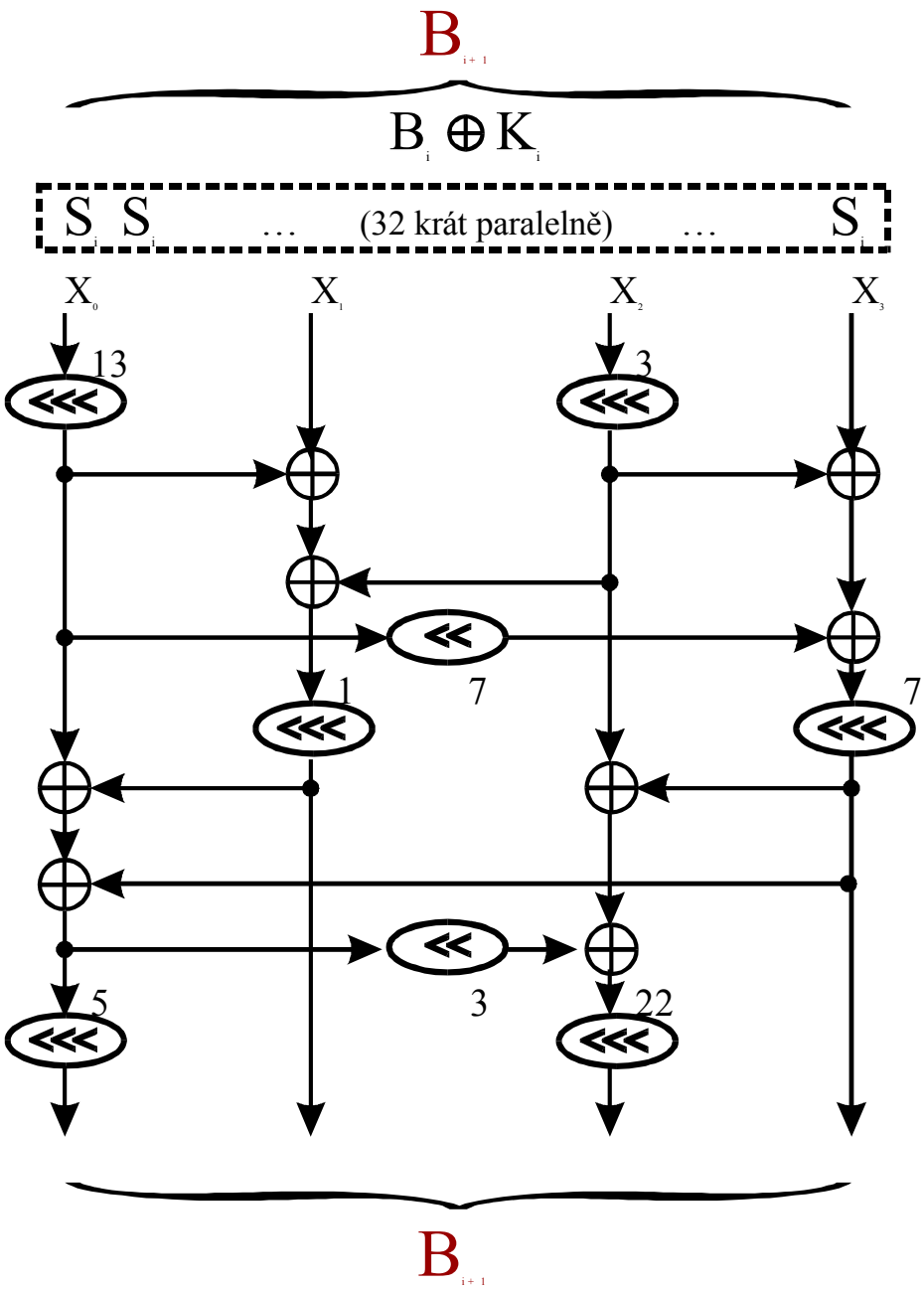
- nezbytné zvýšit odolnost vůči diferenční a lineární kryptoanalýze (maskování klíčem, zvýšení počtu kol)
- nově je třeba reagovat na neteoretické fyzikální útoky na procesor realizující šifru (výběr operací ekvivalentně zatěžujících procesor)
- zvýšení efektivity operací využitím plné délky slova procesoru v jednotlivých operacích

Kryptosystém Serpent

32 kol SP síť, vstup 128 bit plaintext, výstup 128 bit šifra, klíč variabilní délky až 256 bitů
konzervativní návrh využívající konstantní bitové rotace, substituce, XOR
součástí definice algoritmu 8 konstantních S-boxů S_0 … S_7 se vstupem a výstupem o šíři 4 bity

Šifrování

Provede se 31 kol následujícího procesu,
na závěr algoritmus provede ještě jedno míchání s klíčem, substituci a závěrečné míchání s klíčem



Tvorba subklíčů

doplnit klíč K na 256 bitů (přidáním 1000…0b)

1. $K = w_{-8}, w_{-7}, w_{-6}, w_{-5}, w_{-4}, w_{-3}, w_{-2}, w_{-1}$
2. $w_{-i} = (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{-1} \oplus \phi \oplus i) <<< 11$ kde $i \in 0, 1, \ldots, 131$
3. $\{k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}\} = S_{(i \bmod 8)+3}(w_{4i}, w_{4i+1}, w_{4i+2}, w_{4i+3})$ kde $i \in 0, 1, \ldots, 33$
4. $K_i = \{k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}\}$

Dešifrování

Spočte se inverzní hodnota pro všechny S-boxy a algoritmus se spustí „pozpátku“ přičemž všechny operace se invertují

Analýza

Algoritmus je navrhován aby byl odolný proti známým metodám analýzy, byl jedním z pěti postupujících kandidátů pro AES. Bylo publikováno několik útoků omezujících v malé míře bezpečnost šifry, např kompromitace 6 – 11 kol algoritmu vesměs typu known plaintext w potřebou 2^{110} plaintextů a složitostí cca 2^{200} proti neoslabenému algoritmu.

Kryptosystém *Rijndael*

produkční bloková šifra s proměnnou délkou bloku 16, 24 nebo 32 bajtů a klíčem o délce 128, 192, 256 bitů
založena na podobném principu jako algoritmus Square
Stavem šifry označujeme obsah pole $a_{i,j}$ o rozměrech 4 x (délka_bloku / 32)
Podobně klíč je pole $k_{i,j}$ o rozměrech 4 x (délka_klíče / 32)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{0,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,0}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{0,3}$
$k_{2,0}$	$k_{2,0}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

blok plaintextu je do stavu vkopírován v pořadí $a_{0,0}$, $a_{1,0}$, atd. podobně klíč
počet kol je závislý na délce bloku a klíče:

Klíč / Blok	4	6	8
4	10	12	14
6	12	12	14
8	14	14	14

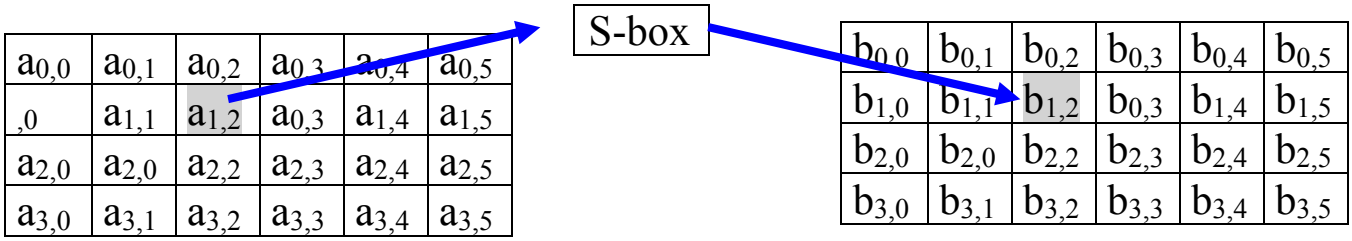
Šifrování

```
Round(State,RoundKey){
    ByteSub(State);
    ShiftRow(State);
    MixColumn(State);
```

```
        AddRoundKey(State,RoundKey);
}

FinalRound(State,RoundKey){
        ByteSub(State) ;
        ShiftRow(State) ;
        AddRoundKey(State,RoundKey);
}
```

Zde:
ByteSub(State)



S-box je nelineární invertibilní transformace definovaná ve dvou krocích –

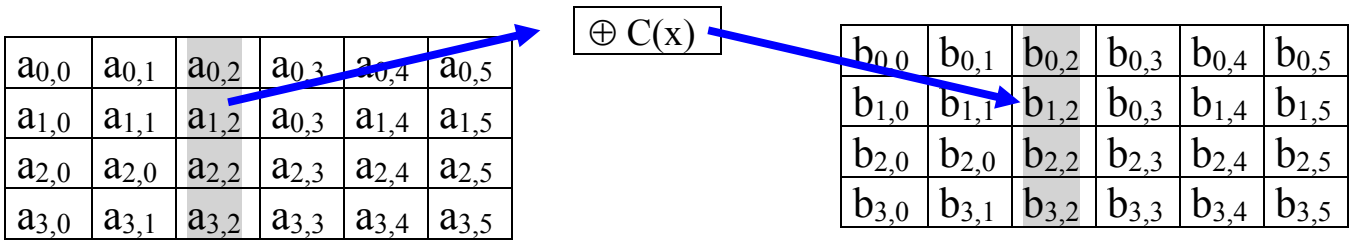
- provede se invertování (vůči násobení) nad GF(2^8) , 0 je inverzní sama k sobě
- aplikuje se následující transformace

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

ShiftRow (State)
provádí rotaci řádků 1, 2 a 3 stavu o pevnou hodnotu závislou na velikosti stavu

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

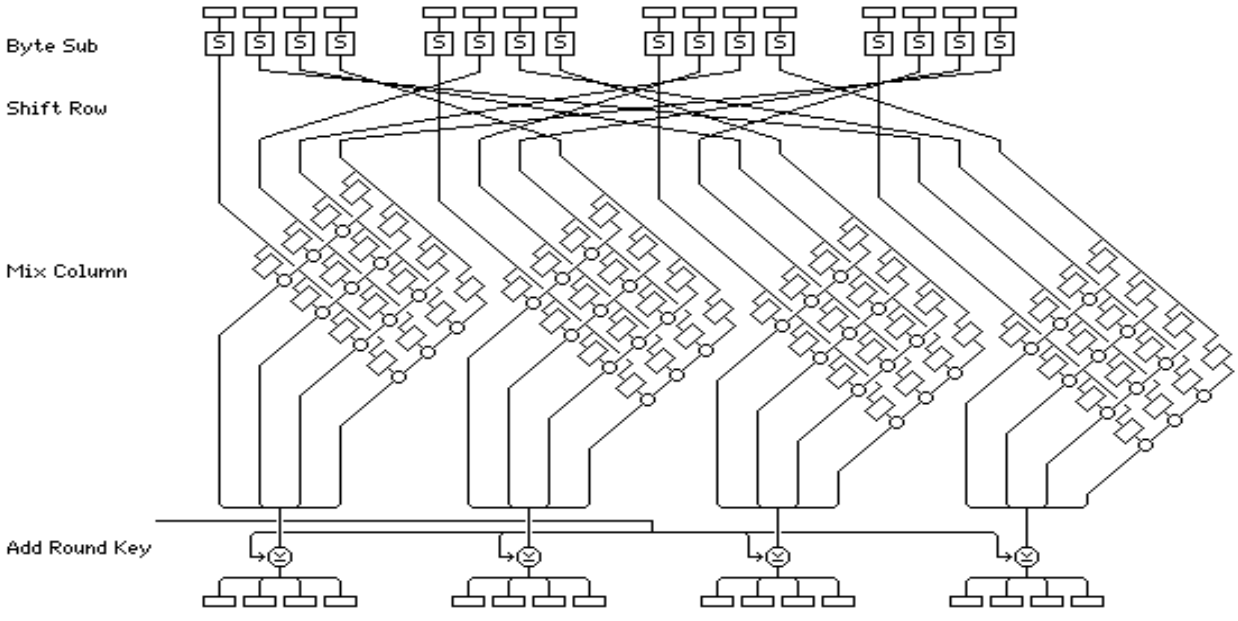
MixColumn (State)



realizuje násobení sloupce jako polynomu nad GF(2^8) konstantním polynomem $C(x) = 3x^3+x^2+x+2$ modulo x^4+1

AddRoundKey (State, RoundKey)
provádí míchání (XOR) stavu s příslušným podklíčem

Zajímavé je, že celé kolo šifrovacího procesu lze na 32 bitovém procesoru implementovat jako 4 výběry z tabulky a 4 XOR operace



Expanze klíče

KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)])
{
for(i = 0; i < Nk; i++)

```
W[i] = (key[4*i],key[4*i+1],key[4*i+2],key[4*i+3]);
for(i = Nk; i < Nb * (Nr + 1); i++)
{
	temp = W[i - 1];
	if (i % Nk == 0) {
		temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
	} else if (i % Nk == 4) {
		temp = SubByte(temp);
	W[i] = W[i - Nk] ^ temp;
	}
}
```

Analýza

Algoritmus byl podroben rozsáhlé analýze a zvolen jako nový standard AES
V současnosti není známa jakákoliv prakticky použitelná slabina
Zvláštností je matematický model prováděných transformací
Publikovány různé útoky do určité míry omezující bezpečnost algoritmu. Původně publikovaný XSL útok není funkční. Publikovány útoky na key schedule se složitostí 2^{99}, ovšem s malou praktickou využitelností. Publikován útok na plnou verzi algoritmu 4x rychlejší než brutal force.

Kryptosystém *Camellia*

publikován v letech 2000 – 2001, doporučen v rámci výzvy NESSIE a japonské CRYPTREK
Feistelova síť, blok o velikosti 128 bitů, klíč 128, 192, 256 bitů
18 kol šifrování s vloženou zvláštní funkční fází po 6 a 12 kole a randomizační XOR operací před prvním a po posledním kole

Šifrování:

randomizace: $P \otimes (kw_1|kw_2) = L_0|R_0$
následuje 18 kol šifrování pro128 bitový klíč, resp. 24 pro ostatní velikosti klíče:
pro r = 1, …5, 7,…,11,13,…18 resp. r = 1, …5, 7,…,11,13,…17, 19, …24

$$L_r = R_{r-1} \otimes F(L_{r-1}, k_r)$$

$$R_r = L_{r-1}$$

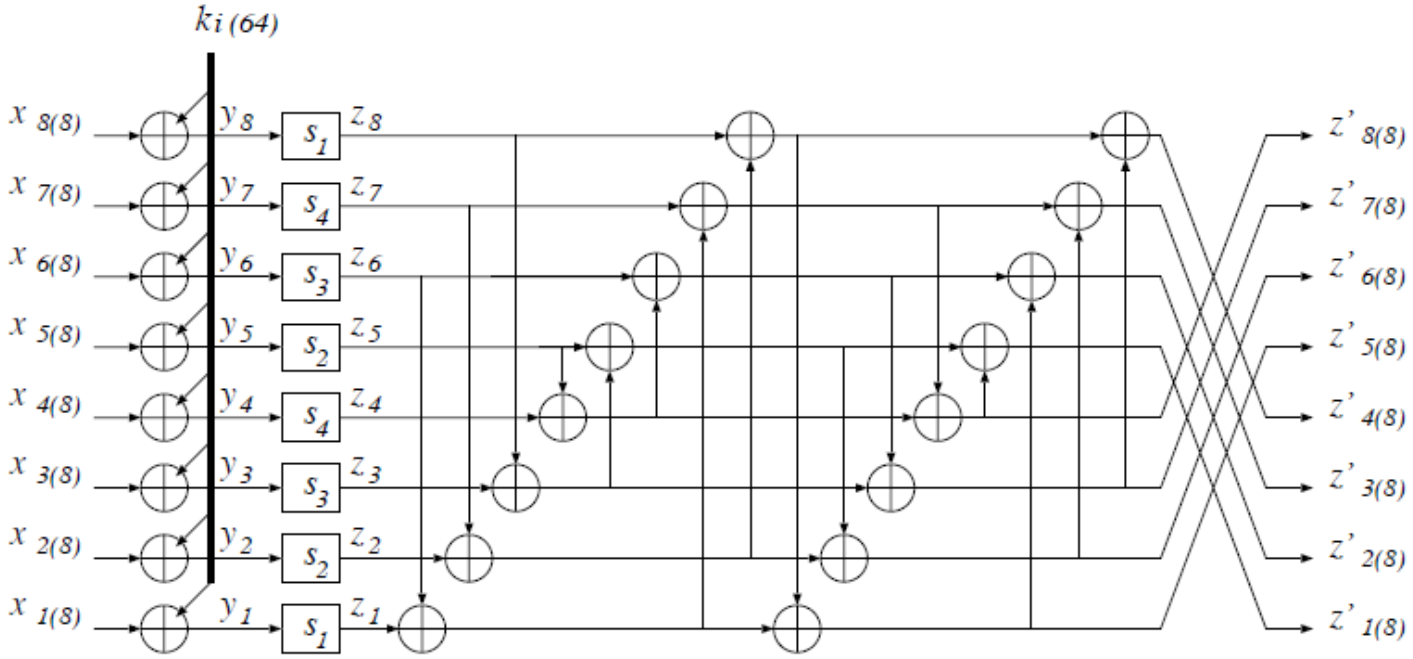
pro $r = 6$ a 12, resp. $r = 6$, 12 a 18

$$L'_r = R_{r-1} \otimes F(L_{r-1}, k_r)$$
$$R_r{}' = L_{r-1}$$
$$L_r = FL\left(L'_r, kl_{2r/6-1}\right)$$
$$R_r = FL^{-1}\left(R'_r, kl_{2r/6}\right)$$

a na závěr ještě jednou randomizace

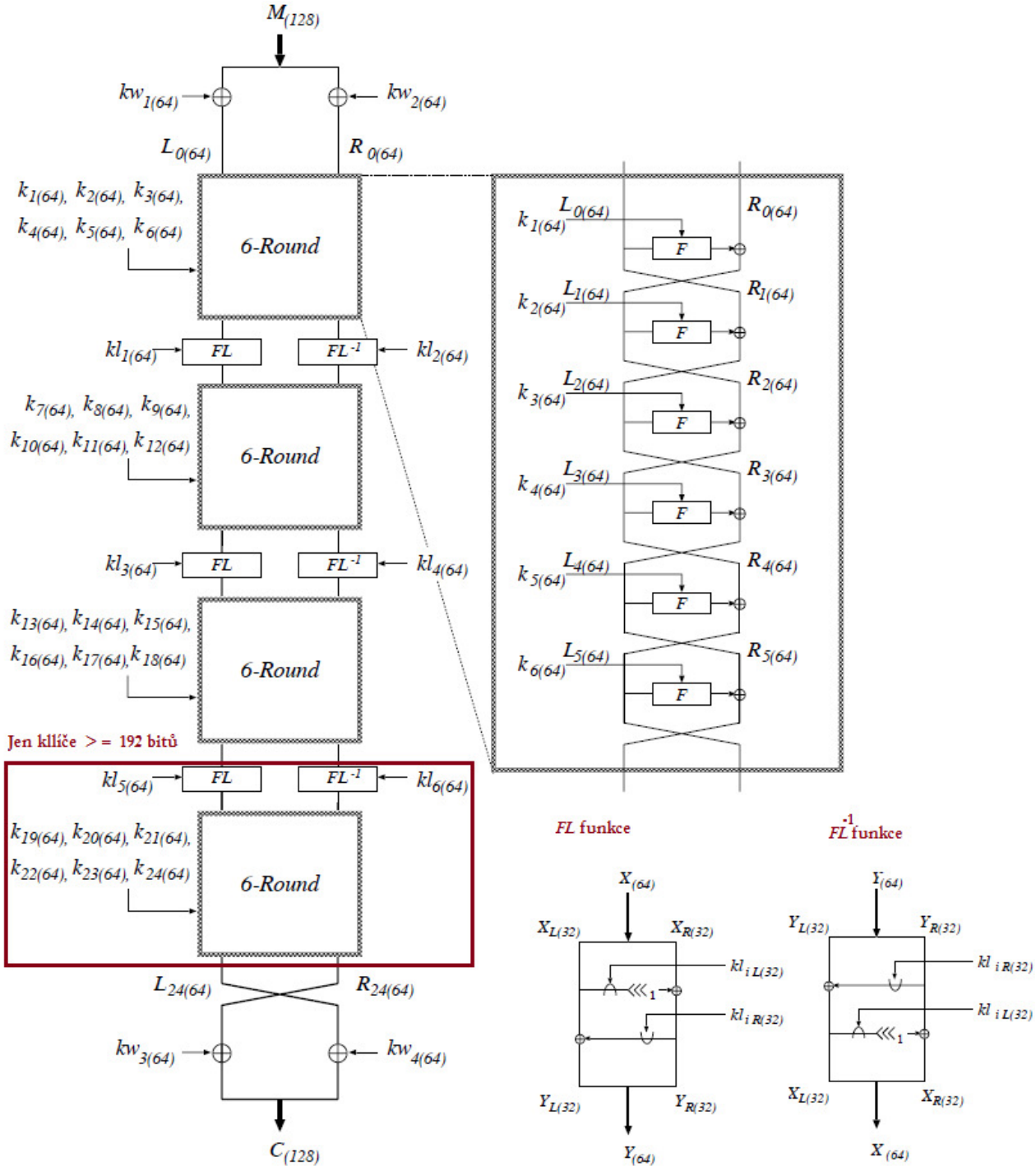
$$C = (R_{18}|L_{18}) \otimes (kw_3|kw_4)$$

F funkce vypadá následovně:



Nejprve se realizuje XOR s klíčem na kolo, následuje 8 paralelně aplikovaných 8-bitových S-boxů a dále promíchání jednotlivých bitů pomocí XOR a permutace
S-boxy jsou statické, definovány algebraicky

Celý algoritmus:



Dešifrování:

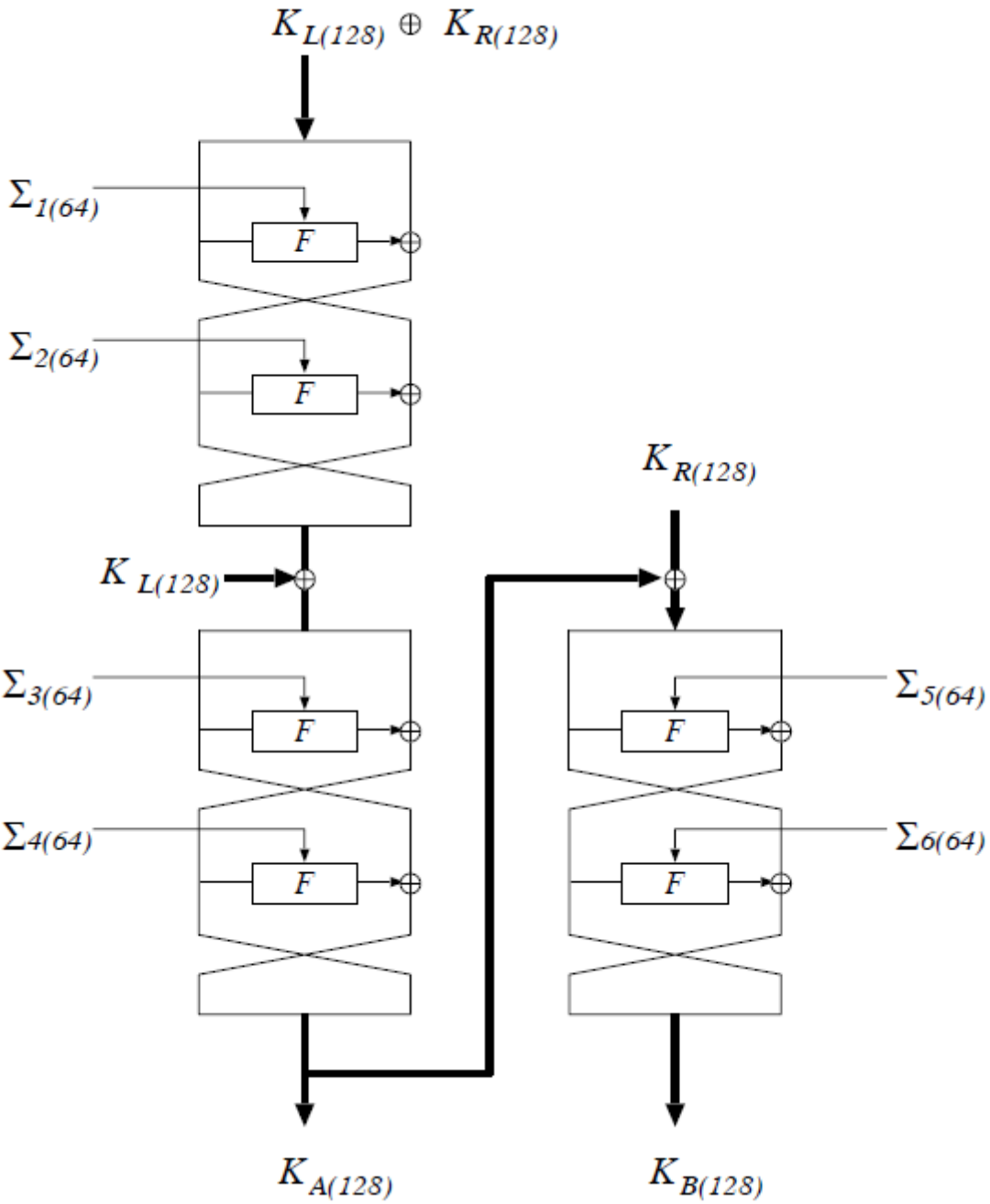
Stejný algoritmus, klíče na kolo se berou v opačném pořadí

Expanze klíče

klíč K se vkopíruje do proměnných K_L a K_R takto:

- 128 bitů - $K_L = K,\ K_R = 0$

- 192 bitů - $K_L = K,\ K_{RL} = K_{(129\text{-}192)},\ K_{LL} = \overline{K_{RL}}$
- 256 bitů - $K_L = K_{(1\text{-}128)},\ K_R = K_{(129\text{-}256)}$



Analýza

Aktuálně se jedné o relativně nový algoritmus s pozitivním hodnocením bezpečnosti, ovšem ani zdaleka neprošla tekovou mírou analýzy, jako AES.

Režimy činnosti blokových šifer

1. *ECB* (electronic code book) - pouze šifrování klíčů

$$C = \mathbf{E}\left(K, P\right)$$

2. *CBC* (cipher block chaining) - vhodný pro šifrování zpráv

$$C_n = \mathbf{E}\left(K, P_n \ \mathbf{xor}\ C_{n-1}\right)$$

3. *CFB* (cipher feed back) - pro šifrování podobné proudovým šifrám

$$C_n = P_n \ \mathbf{xor}\ \mathbf{E}\left(K, \mathbf{shl}\left(Reg, k\right) + \mathbf{left}\left(C_{n-1}, k\right)\right)$$

4. *OFB* (output feed back) - pro aplikace, kdy je třeba eliminovat šíření přenosových chyb, vysokokapacitní spoje s velkou redundancí (řeč, video)

$$H_n = \mathbf{E}\left(K, \mathbf{shl}\left(Reg, k\right) + \mathbf{left}\left(H_{n-1}, k\right)\right)$$

$$C_n = P_n \ \mathbf{xor}\ H_n$$

Proudové šifry

zpracovávají otevřený text po jednotlivých bajtech

Kryptosystém *RC4 (arcfour)*

proudová šifra od R. Rivesta, velmi jednoduchý a rychlý algoritmus
používá stavové pole *S* velikosti 256 bajtů (a ještě jedno pro inicializaci klíče) a dva čítače

Inicializace

pole *S* naplníme čísly 0 – 255 (*S*[0] = 0, *S*[1] = 1, …)
pomocné pole *S2* naplníme klíčem *S*[*i*] = *key*[*i* mod keylen]
zamícháme pole *S*:

j = 0;

```
for (i = 0; i < 256; i++) {
    j += S[i] + S2[i] mod 256;
    S[i] ↔ S[j];
}
```

pole *S2* a proměnné *i*, *j* smažeme

Šifrování

$i = i + 1 \bmod 256;$
$j = j + S[i] \bmod 256;$
$S[i] \leftrightarrow S[j];$
output $S[\ S[i] + S[j] \bmod 256];$

output se míchá s plaintextem pomocí XOR

Analýza

Nedostatkem špatné statistické vlastnosti prvních cca 100 bajtů výstupu
po roce 2012 byly publikovány nové statistické chyby prvních 256 bajtů výstupu, pokud útočním může realizovat šifrování zprávy řádově desítkami milionů různých klíčů, lze zjistit některé bajty původního plaintextu.

Systém *Fish*

proudová šifra založená na Fibonacciho generátoru pseudonáhodných čísel

Vypouštějící generátory (shrinking generators)

předp. dva generátory pseudonáhodných čísel *A* a *S*.

A generuje posloupnost $a_0, a_1, \ldots$ prvků $GF(2)^{n_A}$,

S obdobně posloupnost $s_0, s_1, \ldots$ prvků $GF(2)^{n_S}$

dále budeme používat funkci **d**: $GF(2)^{n_S} \rightarrow GF(2)$

Vypouštějící procedura ponechá k dalšímu zpracování prvky a_i a s_i pokud

$$\mathbf{d}(s_i) = 1$$

aplikací vypouštěcí procedury získáme posloupnosti $z_0, z_1, \ldots$ z $a_0, a_1, \ldots$, resp. $h_0, h_1, \ldots$ z $s_0, s_1, \ldots$

Popis algoritmu Fish

zvolíme n_A, n_S rovno 32
jako A i S budeme používat uzavřený Fibonacciho generátor

$$a_i = a_{i-55} + a_{i-24} \mod 2^{32}$$

$$s_i = s_{i-52} + s_{i-19} \mod 2^{32}$$

Samozřejmě prvky $a_{-55}, a_{-54}, \ldots, a_{-1}$ musí být vhodným způsobem odvozeny z klíče. Obdobně pro posloupnost s_i.
Funkce **d** mapuje 32-bitový vektor na jeho nejméně význačný bit.

Není bezpečné používat k šifrování přímo posloupnost $z_0, z_1, \ldots$:
s pravděpodobností 1/8 totiž trojice a_i, a_{i-55}, a_{i-24} projde celá do posloupnosti $\{z_i\}$

rozdělíme posloupnost $z_0, z_1, \ldots$ na páry (z_{2i}, z_{2i+1}), $h_0, h_1, \ldots$ na páry (h_{2i}, h_{2i+1}) a vypočítáme výsledné hodnoty r_{2i}, r_{2i+1}:

$$c_{2i} = z_{2i} \oplus \left(h_{2i} \wedge h_{2i+1}\right)$$

$$d_{2i} = h_{2i+1} \wedge \left(c_{2i} \oplus z_{2i+1}\right)$$

$$r_{2i} = c_{2i} \oplus d_{2i}$$

$$r_{2i+1} = z_{2i+1} \oplus d_{2i}$$

Jako tradičně $\oplus$ značí **xor**, $\wedge$ označuje bitové **and**.
Protože h_{2i}, h_{2i+1} mají nejnižší bit jedničkový, je vhodné nastavovat nejnižší bit z_{2i}, z_{2i+1}v závislosti na hodnotě r_{2i}, r_{2i+1}.

Šifrování se provádí např. xorováním výsledné posloupnosti s otevřeným textem.

Analýza

Algoritmus byl publikován koncem roku 1993, nebyl nikdy šířeji používán,
Algoritmus je náchylný na known-plaintext-attack – k prolomení stačí řádově tisíce bitů plaintextu

Trivium

Navržena jako model maximálního zjednodušení streamových šifer

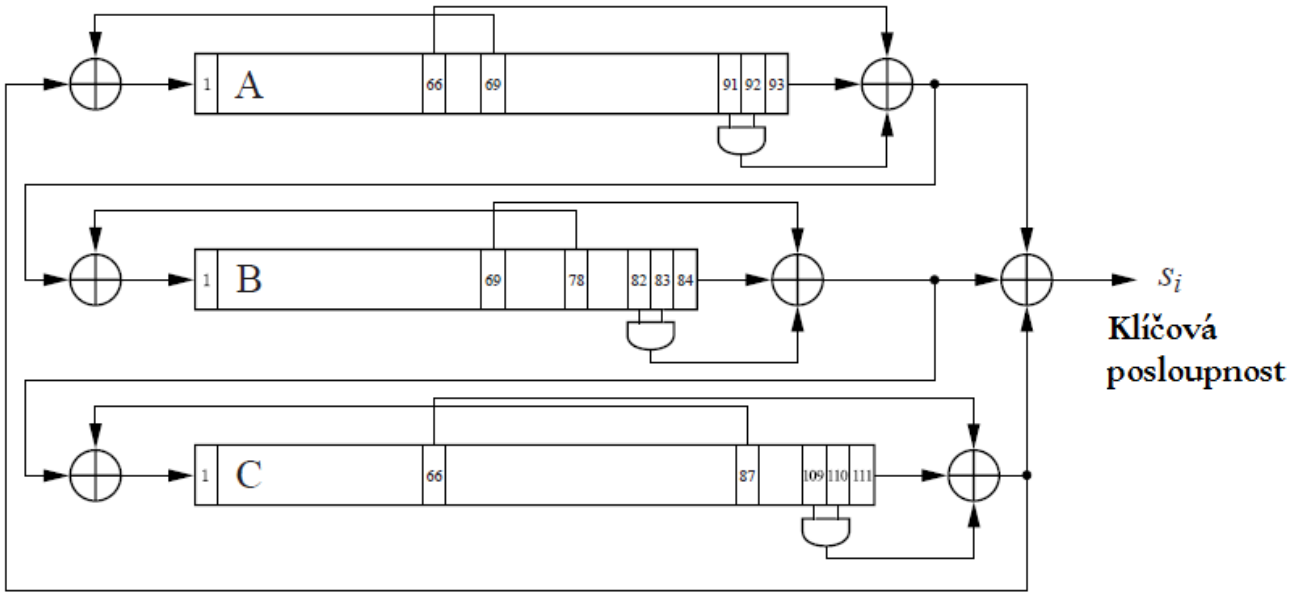
Standard číslo ISO/IEC 29192-3.
v podstatě jde o kombinaci tří posuvných registrů
80 bitový klíč, 80 bitový inicializační vektor

Šifrování

obnova vnitřního stavu

- $a_i = c_{i-66} \oplus c_{i-111} \oplus c_{i-110} \wedge c_{i-109} \oplus a_{i-69}$
- $b_i = a_{i-66} \oplus a_{i-93} \oplus a_{i-92} \wedge a_{i-91} \oplus b_{i-78}$
- $c_i = b_{i-69} \oplus b_{i-84} \oplus b_{i-83} \wedge b_{i-82} \oplus c_{i-87}$

… pozor, toto jsou *bitové* operace



výstupní posloupnost

$r_0 \ldots r_{2^{64}-1}$ je generována

- $r_i = c_{i-66} \oplus c_{i-111} \oplus a_{i-66} \oplus a_{i-93} \oplus b_{i-69} \oplus b_{i-84}$

inicializace klíčem

0-ti bitový klíč $k_0 \ldots k_{79}$ a l-bitový IV $v_0 \ldots v_{l-1}$ (where $0 \le l \le 80$), se použijí takto:

- $(a_{-1245} \ldots a_{-1153}) = (0, 0 \ldots 0, k_0 \ldots k_{79})$
- $(b_{-1236} \ldots b_{-1153}) = (0, 0 \ldots 0, v_0 \ldots v_{l-1})$
- $(c_{-1263} \ldots c_{-1153}) = (1, 1, 1, 0, 0 \ldots 0)$

a spočítá se 1152 inicializačních iterací

I zde $\oplus$ značí **xor**, $\wedge$ označuje bitové **and**.

Dešifrování

:-)

Analýza

v roce 2014 publikována adaptace Cube Attack proti redukované verzi algoritmu s pouze 735 kroky inicializace a složitostí 2^{30} iterací
dosud bezpečný, ale s malou rezervou

Salsa20

Streamová šifra, klíč *k* 16 – 32 bajtů, 8 bajtů nonce *n*
Používá operace XOR (⊗), sčítání mod 32 (+) a pevné rotace (<<<)
V zásadě se jedná o hashovací funkci v counter modu

Šifrování

Stav šifry si nejlépe představit jako 4 x 4 matici 32-bitových slov

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}

Základní primitivou je operace **quarterround**, tj modifikace jednoho řádku stavové matice:
pro $x=(x_0;x_1;x_2;x_3)$
quarterround$(x)=(z_0;z_1;z_2;z_3)$ kde

$$z_1 = x_1 \otimes ((x_0 + x_3) <<< 7);$$
$$z_2 = x_2 \otimes ((z_1 + x_0) <<< 9);$$
$$z_3 = x_3 \otimes ((z_2 + z_1) <<< 13);$$
$$z_0 = x_0 \otimes ((z_3 + z_2) <<< 18)$$

Rowdown funkce není ničím jiným než paralelní aplikací quarterround funkce na vstup délky 16 slov:

Buď $x=(x_0;x_1;x_2;x_3;\ldots;x_{15})$

rowround$(x) = (z_0;z_1;z_2;z_3;\ldots;z_{15})$ kde

$(z_0;z_1;z_2;z_3) =$ **quarterround**$(x_0;x_1;x_2;x_3)$
$(z_5;z_6;z_7;z_4) =$ **quarterround**$(x_5;x_6;x_7;x_4)$
$(z_{10};z_{11};z_8;z_9) =$ **quarterround**$(x_{10};x_{11};x_8;x_9)$
$(z_{15};z_{12};z_{13};z_{14})=$**quarterround**$(x_{15};x_{12};x_{13};x_{14})$

Hashovací funkce Salsa

Salsa20$(x)=x+\text{rowround}^{20}(x)$

provádí transformaci 64bajtového vstupu na 64bajtový výstup
A teď konečně šifrování:
mějme

- Klíč k o délce 32 bajtů se rozdělený na poloviny k_0, k_1,
- n je 8-bajtová nonce náhodně volená pro každou zprávu (např id zprávy)
- t0, t1, t2, t3 jsou pevně definované sekvence.
- i je pořadové číslo bloku (counter)

i-tý blok plaintextu m_i se zašifruje

$c_i =$ **Salsa20**$(t_0;k_0;t_1;n;i;t_2;k_1;t_3) \otimes m_i$

… s tím, že pro šifrování posledního bloku zprávy se použije pouze tolik bajtů klíčové posloupnosti, kolik je třeba. Pro klíč délky 16 bajtů se položí $k_0 = k_1$

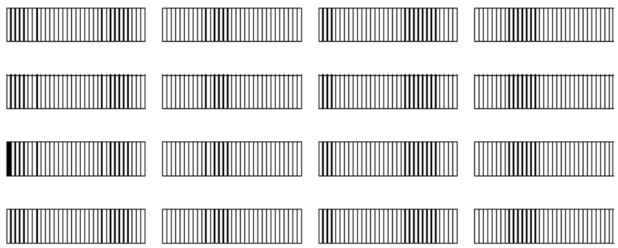
Pozn.: rozumné implementace neztrácejí čas transpozicí stavové matice a namísto toho realizují 10 „dvoukol“ které se skládají z operace ***rowround*** *a její transponované verze – operace* ***rowcolumn***

Analýza

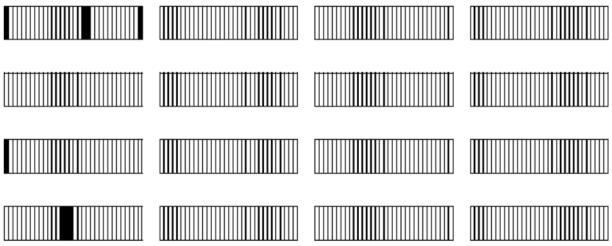
Algoritmus je v současné době považován za bezpečný, stal se vítězem výzvy eSTREAM v roce 2005, IETF zvažuje jeho standardizaci pro TLS a další použití

Pro zajímavost: šíření změny jediného bitu stavem algoritmu v průběhu výpočtu:

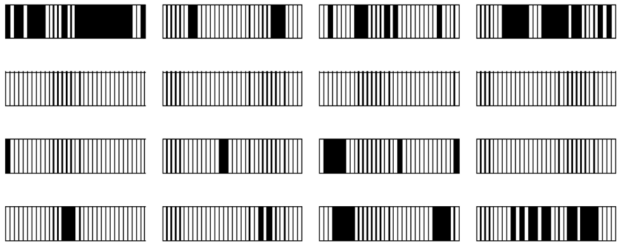
vstupní diference



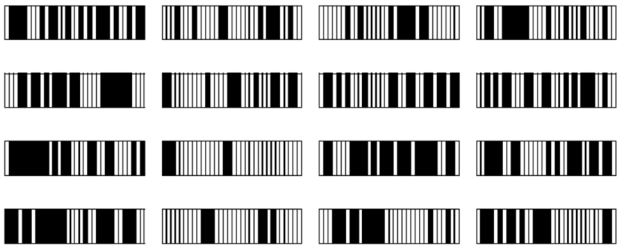
po prvním kole



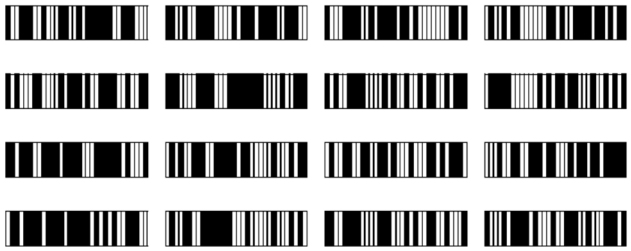
po druhém kole



po třetím



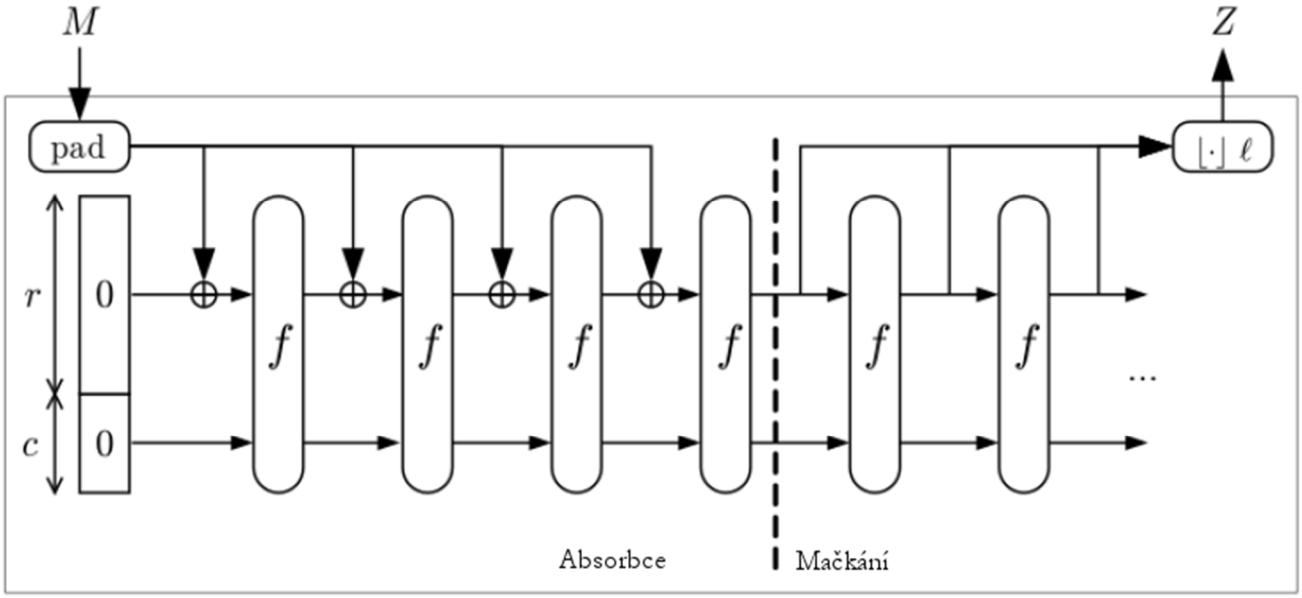
a po čtvrtém



do konce zbývá ještě 16 kol!

Keyak

symetrický kryptografický algoritmus zajišťující zároveň šifrování a autentizaci zprávy
založen na principu struktury sponge



sponge

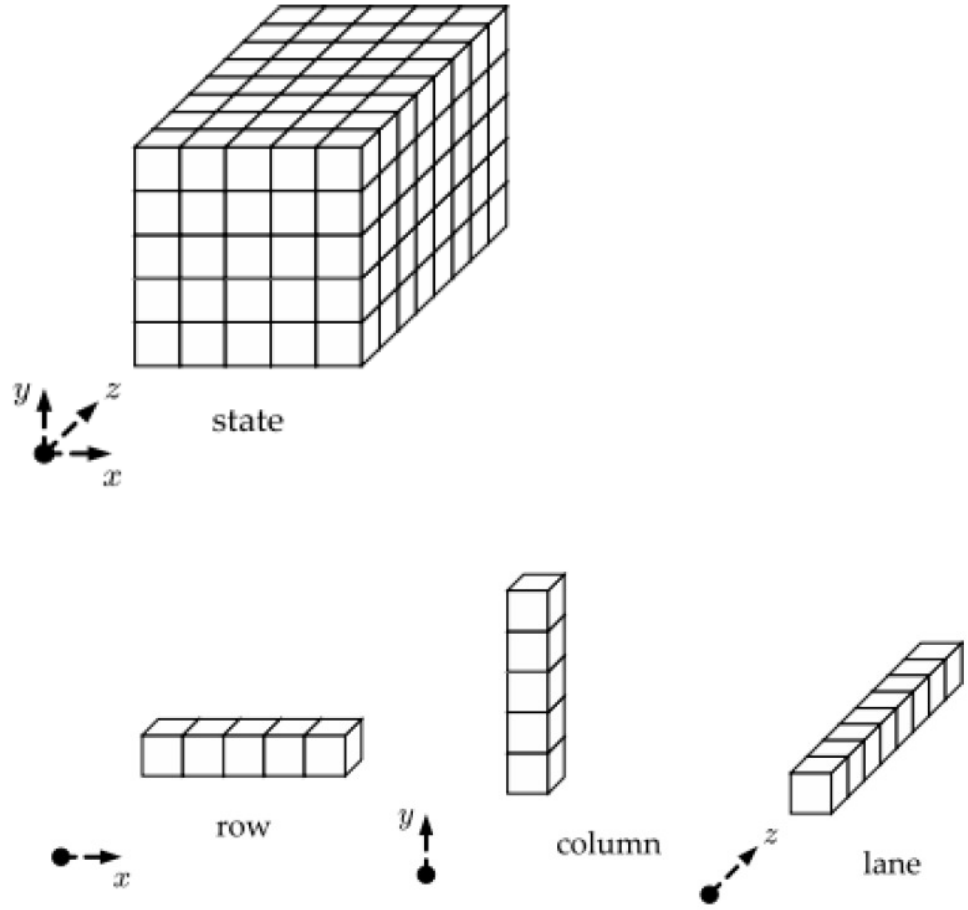
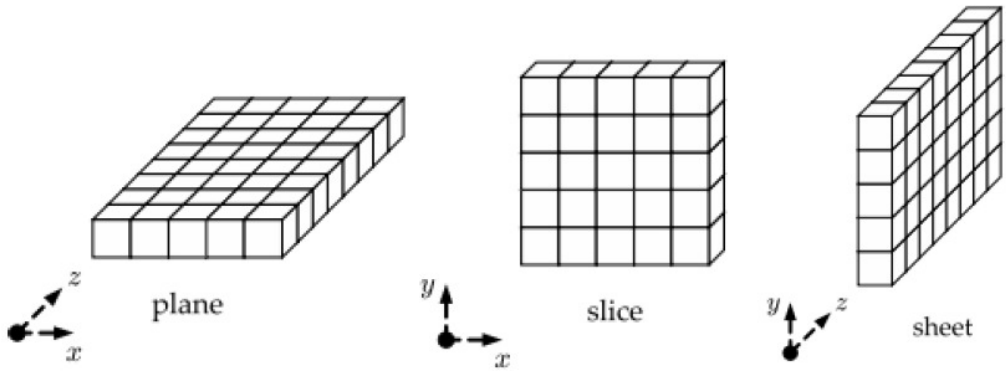


Fig. 6b: *Keccak$_f$ state row, column and lane* [21]



```
Round[b](A, RC):
  //θ-step for bit diffusion
  for x ∈ {0,..,4}:
    C[x] = A[x,0] xor A[x,1] xor A[x,2] xor A[x,3] xor A[x,4]

  for x ∈ {0,..,4}:
    D[x] = C[x-1] xor rot(C[x+1], 1)

  for (x, y) ∈ {{0,..,4}x{0,..,4}}:
    A[x,y] = A[x,y] xor D[x]

  //ρ-step for inter-slice diffusion and
  //π-step for disturbing x,y alignment through lane-transposition
  for (x, y) ∈ {{0,..,4}x{0,..,4}}:
    B[y, 2x + 3y] = rot(A[x,y], r[x,y])

  //χ-step for non-linear mapping
  for (x, y) ∈ {{0,..,4}x{0,..,4}}:
    A[x,y] = B[x,y] xor ((not B[x+1,y]) and B[x+2,y])

  //ι-step to break symmetry
  A[0,0] = A[0,0] xor RC

  Return A
```