DES

přednáška Úvod do kryptologie ($\varphi - \varepsilon$ verze)

Michal Havlíček

KM FJFI ČVUT

28. března 2011

Def: **šifrovací funkce** je bijekce $E_e : M \to C$,
$e \in K$ klíč jednoznačně určující E_e (podobně dešifrovací $D_d : C \to M$).
Def: **kryptosystém** (šifra) - systém množin $\{E_e : e \in K\}$ (šifrování) a
$\{E_e^{-1} : e \in K\} = \{D_d : d \in K\}$ (dešifrování)
tj. $\forall e \in K \ \exists$ jednoznačné $d \in K$:
$D_d = E_e^{-1}$: $D_d(E_e(m)) = m \quad \forall m \in M$, (e, d) - key-pair.

Def: **bloková šifra** je kryptosystém rozdělující otevřený text na řetězce (tzv. **bloky**) pevné délky $k \in N$ (**délka bloku**) a každý blok je šifrován jako celek najednou.
Def: **transpoziční (permutační) šifra** (jednoduchá transpoziční, někdy jednoduchá permutační) je bloková šifra se symetrickým klíčem, s délkou bloku $r \in N$, s prostorem klíčů $K =$ množina permutací $\hat{r}$ (tj. $|K| = r!$),
$E_e(m) = (m_{e(1)}, m_{e(2)}, \ldots, m_{e(r)}), \ \forall m = (m_1, m_2, \ldots, m_r) \in M, \ e \in K$
$D_d(c) = D_{e^{-1}}(c) = (c_{d(1)}, c_{d(2)}, \ldots, c_{d(r)}), \ \forall c = (c_1, c_2, ..., c_r) \in C$.
pozn: permutační šifry nemění četnosti jednotlivých znaků.

Def: (**substituční šifra**) A - n prvková abeceda, M - množina všech bloků délky r nad A, prostor klíčů K je množina všech uspořádaných r-tic:
$e = (\sigma_1, \sigma_2, ..., \sigma_r)$ permutací na A,

$\forall e \in K$ a $m = (m_1, m_2, ..., m_r) \in M$:
$E_e(m) = (\sigma_1(m_1), \sigma_2(m_2), ..., \sigma_r(m_r)) = (c_1, c_2, ..., c_r) = c \in C$

$\forall d = (d_1, d_2, ..., d_r) = (\sigma_1^{-1}, \sigma_2^{-1}, ..., \sigma_r^{-1}) = \sigma^{-1}$:
$D_d(c) = (d_1(c_1), d_2(c_2), ..., d_r(c_r)) = (\sigma_1^{-1}(c_1), \sigma_2^{-1}(c_2), ..., \sigma_r^{-1}(c_r)) = m$

pak tuto šifru nazveme substituční (pokud všechny klíče stejné pak jde o tzv. jednoduchou substituční šifru (**monoalfabetická**), jinak tzv. **polyalfabetická**).
pozn: na monoalfabetickou (Caesar) lze aplikovat frekvenční analýzu! na polyalfabetickou (Vigenere) také - pokud známe délku bloku lze text rozdělit do bloků a aplikovat jí na každý blok zvlášť.

zavedl Shannon (1949), proti frekvenční analýze, každá "bezpečná" šifra musí! obsahovat obě tyto techniky.

confusion (zmatení) - maskování vztahu mezi otevřeným a šifrovým textem, znesnadňuje analýzu (nelze hledat žádné vzory a statisticky je zpracovávat), je třeba využít nějaké složité substituce.
diffusion (rozptyl) - rozptyl redundance otevřeného textu přes celý šifrový (aby nebylo možno z redundancí něco dedukovat), např. opakováním permutace s následnou aplikací nějaké funkce.

jde o to, že permutace (která se má tvářit nahodile) bloku je konstruována z několika permutací (podobně "náhodných") menších bloků tak, že klíč specifikuje tyto menší permutace (čímž se právě dosahuje "zmatení"):
$F_k(x) = f_1(x_1)f_2(x_2)\ldots$
problém: permutace nejsou zcela náhodné, řešení: přeházení bitů ("rozptyl") aplikované spolu s "zmatením" - **round**, rounda probíhá při šifrování obecně vícekrát (navíc v různých "kolech" mohou být aplikovány jiné permutace, výběr může být obecně závislý na klíči).
pozn: monoalfabetická substituce ani jedno, polyalfabetická (např. Vigenere) pouze "zmatení".

S-BOX (substituční box) představuje pevně danou malou permutaci f_i, nezávislý na klíči, zásadně ovlivňuje výslednou "kvalitu" šifry.

rozumné požadavky na S-BOX:
1) musí jít o bijektivní zobrazení (jinak by nešlo o permutaci!), (naopak platí: když boxy jsou bijekce pak: nehledě na systém tvorby sub-klíčů a počet kol je F_k permutace).
2) **lavinový efekt** = malá změna vstupu vyvolá velké změny ve výstupu, konkrétně: změna v jednom bitu ovlivní (ne změní!) všechny bity výsledku (ideálně pravděpodobnost změny ≈ 0.5), lze dosáhnout (při rozumném počtu kol) takto: změna jednoho bitu vstupu S-BOXu vede na změnu aspoň dvou bitů výstupu $\longrightarrow$ permutace zajistí že nikdy není za sebou použit stejný S-BOX.

Feistelova síť

alternativní způsob konstrukce blokové šifry, také využívá S-BOXy, permutace.., ale celkový design a hlavní myšlenka jiná, zajímavý důsl: S-BOXy nemusí! být bijekce - velmi dobrá vlastnost - ve výsledku už není tolik vidět systém $\approx$ blíží se nahodilosti.

bloková šifra, vstup dvojice otevřeného textu (L_0, R_0) obě části délky b, výstup dvojice šifrového textu (R_r, L_r), obě části opět délky b.
iterační proces (podle $r \in N$): vstupní klíč k (**master key**) slouží ke generování sub-klíčů k_j, $j \in \hat{r}$ (**key schedule**), obecně: $k_j \neq k_i$ pro $i \neq j$ a $k \neq k_i \ \forall i$.
mangler function (řídící fce): definována pro každé kolo, vstup: sub-klíč a polovina výstupu (předchozího kola).

round function: iterační proces (dle r) aplikovaný na pár $(R_{j-1}, k_j)(j \in \hat{r})$ (pár tedy vystupuje jako otevřený text), výstupem je (šifrový) pár
$(L_j, R_j), \quad L_j = R_{j-1}, \quad R_j = L_{j-1} \oplus f(R_{j-1}, k_j)$ (***)

tj. pokud $(L_0, R_0) = (R_{-1}, R_0)$ na začátku pak:
$(L_{j-1}, R_{j-1}) = (R_{j-2}, R_{j-1})$ je vstup a
$(L_j, R_j) = (R_{j-1}, L_{j-1} \oplus f(R_{j-1}, k_j))$ výstup, $\forall j \in \widehat{(r+1)}$.

v každém kole jsou substituována data nalevo (z předchozí iterace) pomocí $L_{j-1} \oplus f(R_{j-1}, k_j)$ čímž vzniká $(L_{j-1} \oplus f(R_{j-1}, k_j), R_{j-1})$
následováno permutací jejíž výsledkem je $(R_{j-1}, L_{j-1} \oplus f(R_{j-1}, k_j))$.

pozn: pokud je toto schéma bráno jako šifrování pak dešifrování je aplikace stejného schématu se vstupem šifrový text a aplikací sub-klíčů v opačném pořadí, využívá princip confusion+diffusion, ale klíč je využit k jinému účelu. komponenty nejsou invertibilní ale celek ano!

tvrz: funkce klíčů F je definovaná Feistelovou sítí, pak bez ohledu na mangler funkce a počet round: F_k je permutace pro libovolné k.
dk: stačí ukázat, že libovolné rounda je invertibilní,
vstup i-té rundy (L_i, R_i) pak (L_{i-1}, R_{i-1}) lze spočíst: $R_{i-1} = L_i$ a
$L_{i-1} = R_i \oplus f_i(R_{i-1})$,
funkci f_i lze odvodit z mangler fce f a master-key k,
tedy jde o inverzi (***) $\square$.

dodatky: mimo boxů, permutací může síť obsahovat libovolné další funkce (není v rozporu s def), velikost bloku: čím větší tím lepší (původně běžně 64-bit, moderní 128-bit), délka klíče: čím delší tím lepší (původně 56-bit, dnes typicky 128-bit, ale i 256-bit a víc), round function: 16 je tak nejběžnější ($\exists$ i šifry s 32 rounds).

útoky na šifru

vstup: nějaká informace o otevřeném nebo (odpovídajícím) šifrovém textu.
cíl: zjistit klíč.
pasivní útok = kryptoanalytik pouze monitoruje komunikaci (Alice, Bob & Eve).
aktivní útok = navíc komunikaci upravuje (Alice, Bob & Mallory).

pasivní druhy útoků:
chosen-plaintext - vybraný text se zašifruje, porovnáním textů (šifrového a otevřeného) lze zjistit klíč.
chosen-ciphertext - vybraný šifrový text se dešifruje a výsledek se porovnává s šifrovým textem.
known-plaintext - je k dispozici nějaký vzorek otevřených a odpovídajících šifrových, efektivní, stačí málo takových dvojic.
ciphertext-only - je k dispozici šifrový text jakožto způsob jak vydedukovat klíč (systém neodolný vůči tomuto útoku = "velmi" nebezpečný).
adaptive chosen-plaintext - varianta chosen-plaintext kde výběr je ovlivněn předchozím (obdrženým) otevřeným textem (podobně adaptive chosen-ciphertext).

speciální kategorie: metoda hrubé síly = úplné prohledávání prostoru klíčů, mělo by být časově/výpočetně náročné...ale!

DES = **d**ata **e**ncryption **s**tandard
vývoj sedmdesátá léta IMB+National Security Agency (S-BOXy) - impulsem prudké rozšíření počítačů v USA na začátku 70.let.

1973 - ministerstvo obchodu USA vyhlašuje soutěž na tvorbu šifrovacího algoritmu, požadované vlastnosti: bezpečnost (i při neutajení algoritmu), "pochopitelnost", dostupnost, vhodnost pro různé aplikace, výkon, ekonomičnost $\rightarrow$ žádný algoritmus nesplnil...
1974 - druhé kolo - vyhrál DES (odvozen z šifry LUCIFER - interní šifra IBM).

24.2.1975 - patentováno, patent zveřejněn v březnu, bezplatné použití v USA.
23.11.1976 - přijat jako standard.
15.1.1977 - standard zveřejněn.
15.7.1977 - standard účinný + pojmenování DES.
určen pro ochranu civilních dat a vládních institucích (mimo ozbrojených složek), předpokládaná životnost do 15 let, NSA si vymohla utajení některých skutečností z designového návrhu DES $\rightarrow$ vedlo k podezření že S-BOXy jsou navrženy tak, aby NSA mohla luštit cizí šifry $\leftarrow$ vyšetřováno zvláštní komisí senátu v roce 1979.

S-DES

simplified **DES** - zjednodušená verze DES určená pro pedagogické účely.
základní charakteristiky: bloky délky 8 bitů, 10-bit klíč (k), 2 rounds.

nástin algoritmu:
1) na blok otevřeného textu (m) je aplikována počáteční permutace (IP).
2) 8-bit vstup je rozdělen na 4 levé (L) a 4 pravé bity (R).
3) R je rozšířeno na 8-bitů expanzní permutací E.
4) výsledek $E(R)$ je přičten modulo 2 k 8-bit sub-klíči SK (generovaného z k).
5) výsledek $E(R) \oplus SK$ je rozdělen na 4-bit bloky (levé a pravé bity) L_1, R_1.
6) L_1 resp R_1 vstupují do S-BOXu S_1 resp S_2 (každý box vrací 2-bit blok).
8) výstupy S-BOXů formují 4-bit blok (L_1*, L_2*), který je permutován P.
9) výsledek (Z) je přičten modulo 2 k L čímž vzniká $L*$.
10) výstup (roundy) je pak ($L*, R$).
11) levé 4-bity se prohodí s pravými 4-bity.
následuje druhá aplikace kroků 3-11.
výstup je permutován IP^{-1} výsledkem je šifrový text.

$IP = (2, 6, 3, 1, 4, 8, 5, 7)$ *př:* $m = (10010111)\ IP(m) = (01011101)$

$EP = (4, 1, 2, 3, 2, 3, 4, 1)$ *př:* $x = (1001)\ EP(x) = (11000011)$

generování sub-klíčů: 10-bit klíč (k) slouží ke generování (2) sub-klíčů,
nejprve je na klíč k aplikována permutace $P_{10} = (3, 5, 2, 7, 4, 10, 1, 9, 8, 6)$,
následuje cyklický posun vlevo o jeden bit ($LS1$) aplikovaný na 5-bit bloky
tvořené bity klíče $(e_3 e_5 e_2 e_7 e_4)$ resp $(e_{10} e_1 e_9 e_8 e_6)$,
následuje selektivní permutace P_8 (vybere jen určité bity 8 z 10 a ty permutuje):
$P_8(e_5 e_2 e_7 e_4 e_3 e_1 e_9 e_8 e_6 e_{10}) = (e_1 e_7 e_9 e_4 e_8 e_3 e_{10} e_6) = k_1$ (sub-klíč 1),
vyjdeme opět z $(e_5 e_2 e_7 e_4 e_3 e_1 e_9 e_8 e_6 e_{10})$ ale tentokrát na 4-bit podbloky
aplikujeme cyklický posun o dva bity vlevo ($LS2$), výsledkem je
$(e_7 e_4 e_3 e_5 e_2 e_8 e_6 e_{10} e_1 e_9)$, které po selektivní permutaci P_8 dá k_2 (sub-klíč 2).

S_1	x_2	0	0	1	1
	x_3	0	1	0	1
x_1	x_4				
0	0	01	00	11	10
0	1	11	10	01	00
1	0	00	10	01	11
1	1	00	01	11	10

S_2	x_2	0	0	1	1
	x_3	0	1	0	1
x_1	x_4				
0	0	01	00	10	11
0	1	10	00	01	11
1	0	11	00	01	10
1	1	10	01	00	11

jak tohle funguje?: 4-bit vstup S-BOXu ($x_1x_2x_3x_4$) určuje, který řádek a sloupec boxu (matice) je vybrán, hodnota tohoto prvku (2 bitová) je pak výstup.(y_1y_2).

round fce: EP(4-bit vstupu x) a součet modulo 2 se sub-klíčem $SK = y = (y_1y_2y_3y_4y_5y_6y_7y_8)$, rozdělení na bity napravo a nalevo:

$L(y) = (y_1y_2y_3y_4)$, $R(y) = (y_5y_6y_7y_8)$, $L(y)$ je vstup S_1 a $R(y)$ je vstup S_2:

$(S_1(L(y) = (z_1z_2)$, $S_2(R(y) = (z_3z_4))$, složením vznikne $z = (z_1z_2z_3z_4)$,

na z se aplikuje permutace $P_4 = (2, 4, 3, 1)$, $(P_4(z) = Z)$ tj. $F(x, SK) = Z$,

pak round fce $f_{SK}(t) = (L(t) \oplus F(R(t), SK), R(t))$ kde t je otevřený text.

switch/swap (SW): $m = (L(m), R(m))$, $SW(m) = (R(m), L(m))$

inverzní IP: $IP^{-1} = (4, 1, 3, 5, 7, 2, 8, 6)$

šifrování:
1) na otevřený text m aplikuj IP
2) aplikuj fci f_{k_1} na výsledek 1)
3) aplikuj SW na výsledek 2)
4) aplikuj f_{k_2} na výsledek 3)
5) aplikuj IP^{-1} na výsledek 4)

dešifrování:
1) aplikuj IP na šifrový text c
2) aplikuj f_{k_2} na výsledek 1)
3) aplikuj SW na výsledek 2)
4) aplikuj f_{k_1} na výsledek 3)
5) aplikuj IP^{-1} na výsledek 4)

příklad: $m = (10100101)$, $k = (0010010111)$ $\quad c = (00110110)$

DES

základní charakteristiky: 64-bit bloky, 56-bit klíč (ze kterého se generuje 16 48-bit sub-klíčů), 16 round fcí f_{k_j},

$$c = (IP^{-1} \circ f_{k_{16}} \circ SW \circ f_{k_{15}} \circ SW \circ \ldots \circ f_{k_1} \circ IP)(m)$$

tj. všechny mangler fce stejné, $f_i(R) = \hat{f}(k_i, R))$, DES je Feistelova šifra s počtem rund $r = 16$,

8 (4x16) S-BOXů, $S_j(m_1 m_2 m_3 m_4 m_5 m_6)$ vrací hodnotu z řádku $(m_1 m_6)$ a sloupce $(m_2 m_3 m_4 m_5)$, permutace P_{32}.

key-schedule: každý sub-klíč k_i je permutace 48-bitů z master-klíče (ten je rozdělen na půl a v každém kole je 24-levých-bitů sub-klíče bráno jako jistá podmnožina 28-levých-bitů master-klíče - a stejně pravé-bity sub-klíče) (vše je veřejné!), mangler function: spolu s i-tým sub-klíčem určuje i-tou round fci.

detailní popisu designu DES - až roku 1994 (Coppersmith).
důležitý aspekt: nelinearita!, kdyby S-BOX realizoval pouze lineární fci $\rightarrow$ celé schéma (šifra) je lineární fce tj. všechny bity šifrového textu by byly pouze lineární kombinací bitů otevřeného textu a klíče tj. rozluštění = řešení soustavy lineárních rovnic.
Def: **lineární šifra** je šifra splňující: každý výstupní bit je lineární kombinací vstupních bitů.
takové šifry jsou snadno prolomitelné pomocí known-plaintext útoku:
$c = me$ (matice šifrového textu c, otevřeného m, klíčů e) pak: $e = m^{-1}c$ (čímž získáváme klíč!)
zajištění nelinearity pomocí S-BOXů (požadavky na S-BOXy):
1) žádný výstupní bit nesmí být lineární funkcí vstupních bitů.
2) každý řádek boxu by měl obsahovat všechny možné výstupní kombinace bitů.
3) pokud se dva vstupy liší právě v jednom bitu nebo právě ve dvou prostředních pak se výstupy musí lišit minimálně ve dvou bitech.
4) pokud se dva vstupy liší v prvních dvou bitech a mají stejné dva poslední pak výstupy se musí lišit zcela.
a spousta dalších.., viz D. Coppersmith: "The Data Encryption Standard (DES) and its strength against attacks".

obecná poznámka (důležitá): je velmi složité navrhnout "dobrý" S-BOX - mírně odlišný S-BOX může vést k velmi slabé šifře!
nezkoušejte si doma vytvářet svoje vlastní "neprolomitelné" verze DES, výsledkem může být DĚS...

DES S-BOXy zobrazují 6-bit řetězec na 4-bit řetězec, tabulka 4 řádky, 16 sloupců a každý prvek obsahuje 4-bit řetězec, 6-bit vstup představuje index ($2^6 = 64$) prvku tabulky tak, že první a poslední bit vstupu definuje řádek, zbytek sloupec (hodnota daného prvku pak představuje výstup).

tvrz: S-BOXy DES způsobují lavinový efekt.
"důkaz": dva vstupy lišící se v jednom bitu (např. v levé polovině)
(L_0, R_0), $(L*_0, R*_0)$ (tj. $R_0 = R*_0$),
po prvním kole: (L_1, R_1), $(L*_1, R*_1)$ se stále liší jen v jednom bitu (odlišnost se přesunula z levé na pravou polovinu),
druhé kolo: na pravé strany je aplikovaná mangler fce - pokud není bit, ve kterém se vstupy liší duplikován (při expanzi) liší se výstupy S-BOXů minimálně ve 2-bitech (viz vlastnosti S-BOXu), (L_2, R_2), $(L*_2, R*_2)$ se tedy liší minimálně v 3 bitech (1 rozdíl v levých stranách - což jsou pravé strany minulého kola a 2 jako výsledek tohoto kola),
permutace změní pozice rozdílných bitů (pozn: permutace je vhodně zvolena, náhodná permutace by mohla snížit účinnost šifry!),
další kolo: každý z (dvou) rozdílných bitů vstupuje do jiného S-BOXu, výsledkem je již rozdíl minimálně v 4 bitech na pravé straně (a 2 bitech na levé),
po 7 kolech: všech 32 bitů pravé strany "pozměněno" (tedy po osmém i levé strany),
DES má 16 kol = lavina zaručena! "□".

otevřený text je rozdělen na 64-bit bloky, ty jsou šifrovány 56-bit klíčem (8 bajt slovo = klíč 56-bit + 8-bitů pro kontrolu parity).

počáteční permutace (IP): promixuje všech 64-bitů otevřeného textu, dále je blok "rozdělen" a vstupuje do první roundy jako (L_0, R_0).

expanze (E): rozšíří 32-bitů R_i na 48-bitů:
$E(r_1, ..., r_{32}) = (r_{32}, r_1, r_2, r_3, r_4, r_5, r_4, r_5, r_6, r_7, r_8, r_9, ..., r_{28}, r_{29}, r_{30}, r_{31}, r_{32}, r_1)$

tvorba klíče k_i: klíč (master) k (56-bit) je permutován pomocí $PC1$, výsledek rozdělen na dva 28-bit "registry" C,D (pravé a levé bity),
obsah registrů je v každém kole (round) cyklicky posunut doleva (v round=0,1,8,15 o jeden v ostatních o dva bity),
výsledek je opět spojen dohromady a permutován $PC2$ - tato permutace navíc redukuje výstup na 48 bitů (vynecháním některých bitů)
pozn: 16x48-bitů sub-klíčů obsahuje každý bit klíče (master) k 12-15x.

k výsledku expanze E je (modulo 2) přičten sub-klíč k_i.

výsledek je rozdělen na 8 6-bit částí - ty vstupují do boxů $S1$–$S8$ (jejich výstup je 4-bitový tj. x8 = 32-bit = nová polovina slova pro další round).

výstup S-BOXů (32-bit slovo) je ještě permutován (P).

další krok je: $L_{i+1} = R_i$, $R_{i+1} = L_i \oplus f(R_i, k_i)$ (dle definice Feistel. sítě!).

po 16.kole: prohodí se levé a pravé poloviny a aplikuje se permutace IP^{-1}.

šifrování a dešifrování: stejné schéma jen obrácené pořadí tvorby sub-klíčů $\rightarrow$ výhoda lze využít stejný algoritmus (tj. stejný hardware resp. později software), velmi výhodné ovšem zároveň umožňuje existenci tzv. "slabých klíčů". permutace IP slouží k "rozprostření" vlivu bitů otevřeného textu na všechny ostatní, IP^{-1} při dešifrování tento účinek eliminuje (aby dešifrování bylo opravdu obrácené šifrování, totéž platí pro záměny pravých a levých stran na začátku a na konci).

dle doporučení NIST, (prvně právě pro DES).

ECB (electronic code book)
každý 64-bit blok kódován stejným klíčem.
DES pak představuje kódovou knihu s 2^{64} kódových výrazů (tj. všechny možné varianty otevřeného textu (binární) délky 64 bitů), kde kódové výrazy se nevyhledávají (klasická kódová kniha), ale vypočítávají pomocí klíče (tedy bez znalosti klíče by nemělo být možné slovo dekódovat - proto snaha zamaskovat souvislost: otevřený text, klíč, šifrový text).

šifrování: $E_k(m_j) = c_j$
dešifrování: $E_{k-1}(c_j) = m_j$

problém: stejné bloky otevřeného textu = stejné bloky šifrového - výhoda pro luštitele (nevadí pro krátké zprávy, např zasílání klíče DES, i když se nedoporučuje raději vůbec).

CBC (cipher block chaining)
vstup je součet modulo 2 předchozího 64-bit bloku šifrového textu s následným 64-bit blokem otevřeného textu (výstup jednoho kroku slouží k modifikaci nového vstupu = každý blok šifrového textu závisí navíc na všech předchozích).
změna v m_j vyvolá změny v $c_j, c_{j+1}, \ldots$
IV (inicializační vektor, 64-bitů)
$c_0 = IV$
k - vstupní klíč (64-bit)
otevřený text jako posloupnost 64-bit bloků m_j
šifrování: $c_j = E_k(c_{j-1} \oplus m_j)$
dešifrování: $m_j = E_k^{-1}(c_j) \oplus c_{j-1}$

sčítání modulo 2 odstraňuje problém modu ECB - není jasná (viditelná) souvislost mezi otevřeným a šifrovým textem zřetězení posloupnosti šifrování bloků (chaining).
obecné použití (včetně autentizace: tvorba message authentication code (mac) - elektronický podpis, umožňuje příjemci ověřit si pravost), nejpoužívanější.

problém volby *IV*: znalost *IV* umožňuje realizovat "man-in-the-middle" útok (Bob posílá zprávu Alice, Mallory může získat zprávu dřív než Alice a libovolně jí pozměnit, aniž by o tom Alice a Bob věděli).
neřeší ani konstantní *IV* (to je vlastně výchozí situace problému, protože = ECB) ani znáhodněný *IV* (příjemce musí *IV* znát! + problém skutečného znáhodnění).
možnost: vyjít z one-time-pad (Vernam šifra) = klíč (náhodný) stejně dlouhý jako otevřený text + pouze jednou použit, šifrový text vznikne součtem modulo 2 otevřeného a klíče (jsme v binární abecedě!) (Vernam, Mauborgne, 1918), šifrový text je náhodný = neprolomitelné! (dk: Shannon, 1949).
nonce (number used once) - unikátní číslo, které je využito právě jednou (tj. není třeba ho tajit), *IV* je pak generováno pomocí nonce:
počítadlo (counter, na začátku nastavené na 0) přiřadí zprávě číslo, které slouží ke generování nonce.
nonce je zašifrováno blokovou šifrou, výsledkem je *IV*.
zpráva je zašifrována CBC modem (s tímto *IV*).
příjemci je místo $c_0 = IV$ posláno číslo zprávy.
příjemce nikdy neakceptuje zprávu jíž bylo přiřazeno číslo $\leq$ číslu, které bylo někdy předtím přiřazeno jiné již akceptované zprávě = zabezpečení.

CFB (cipher feedback)
předchozí šifrový text je použit jako vstup, z něj je vytvořen pseudonáhodný řetězec, který je přičten modulo 2 k otevřenému textu, "stream-cipher-oriented (general-purpose) messaging" (zpracovává $n < 64$ bitů najednou = délka zpětné vazby).

$c_0 = IV$
sub-klíče jsou vytvořeny šifrováním předchozího bloku šifrového textu:
$E_k(c_{j-1}) = k_j$
$c_j = m_j \oplus k_j$
(podobně jako CBC, ale c_j závisí na m_k, opět problém IV).

OFB (output feedback)
stejné jako CFB, ale vstup je předchozí výsledek šifrování,
"stream-cipher-oriented (general-purpose) messaging" (zvlášť při požadavku autentizace zpráv).
vstup IV
$k0 = IV$
sub-klíče jsou vytvářeny opakovaným šifrováním IV: $k_j = E_k(k_{j-1})$
pak: $c_j = m_j \oplus k_j$
IV musí být náhodný vektor (výběr nebo generování, viz CBC), klíč není třeba (viz CBC) doplňovat na potřebnou velikost (už jí má) (padding/salting - doplnění bitů v otevřeném textu na potřebný počet).
slabina: stejné IV pro dvě různé zprávy pak lze přičtením šifrového textu modulo 2 získat původní text protože:
$c_i \oplus c_j = m_i \oplus k_i \oplus m_j \oplus k_i = m_i \oplus m_j$
(tj. z rozdílu šifrových lze získat rozdíl otevřených textů, pokud jeden otevřený text máme, lze již zjistit i druhý, navíc i jen z rozdílu lze informaci o otevřeném textu zjistit).
při vazbě menší než 64 je průměrná délka cyklu produkovaného heslamenší než 2^31 tj. při kódování 2^16 b/s= vyčerpání za 18 hodin = vícenásobné použití klíčů - nutno používat vazbu 64.

5) **CTR** (counter)
šifrový text vzniká jako součet modulo 2 otevřeného textu s šifrovaným počítadlem (counter), jehož hodnota se mění s každým blokem (používá se pro přenosy vyžadující vyšší rychlost), návrh 1980, standardizováno (NIST) roku 2001.
nonce n je spojené s počítadlem i a zašifrováno, vzniká klíč: $k_i = E_k(n, i)$
dále: $c_i = m_i \oplus k_i$
počítadlo a nonce tvoří jeden blok (tj. je třeba aby šifra měla dostatečně dlouhé bloky).
využití nonce = bezpečné klíče.
dešifrování: $k_i = E_k(n, i)$, $m_i = c_i \oplus k_i$
spojení counter+nonce = jedinečnost $\longrightarrow$ jedinečné klíče $\longrightarrow$ žádné dva bloky stejné.
výhoda: rychlost (lze paralelizovat tvorbu key-streamu), nevyžaduje padding (stejně jako OFB) (CBC např. ano).
CTR Random Access Property: c_j není třeba dešifrovat pro dešifrování c_{j+1} (toto není možné při "chainingu"!), použití: aplikace zabezpečení síťové komunikace: IPSec.

první kritika (Diffie, Hellman) 1975 - příliš krátký klíč (lze aplikovat útok hrubou silou - ač náročný (relativně) - vždy končí 100% úspěchem.
odhad v té době: vyluštění za den za 10.000$ - test $2^{56} \approx 10^{17}$ klíčů za den tj. $\approx 10^{12}$ klíčů / s - milion čipů testující rychlostí 1M klíčů / s - 1 čip za 10$ = 10.000.000$ náklady - 5 let předpokládaného provozu = 10.000$ náklady na den - ve skutečnosti se na klíč narazí v kratším čase (v polovičním, s pravděpodobností 50%) = snížení ceny za jedno řešení, navíc: ceny čipů neustále výrazně klesají).

doporučení: 128 nebo 256 - bit klíč, nebo vícenásobné šifrování.

konference z 30.-31.8.1976 tyto závěry odmítla (poznámka pod tlustou čarou: konferenci uspořádala NBS, sezvala různé experty z mimovládních organizací, během dopoledne prvního dne se ukázalo, že jen málo z nich opravdu rozumí kryptoanalýze, *"bylo to jako by středověcí fyzikové chtěli kontrolovat výsledky kvantové mechaniky"*).
další kritika poukazuje na komplementárnost a jisté pravidelnosti (nedostatečná nelinearita) v S-BOXech.

slabiny DES

rozumná kritika: veřejný standard by měl mít zveřejněné specifikace, jinak jsou ti co je znají při příp luštění ve výhodě! (úplná specifikace je známa až v roce 1994!)

září 1976 - pracovní setkání matematiků - posouzení kvality DES, nalezeny různé nedostatky, ale nepředložena žádná faktická metoda k zneužití těchto slabin - rozhodnutí ponechat DES bez změn.

další pochybnosti: při vývoji DES NSA "přesvědčila" IBM o vhodnosti! redukovat délku klíče z navrhovaných 128 bitů (délka klíče šifry ze které byl odvozen DES).

Def: **slabý klíč** je klíč k takový, že $E_k(E_k(m)) = m \quad \forall m \in M$
(šifrovací i dešifrovací funkce je tedy stejná).
DES má 4 slabé klíče:
$0^{56}, 1^{56}, 0^{28}1^{28}, 1^{28}0^{28}$
důvod: posun bitů vlevo a operace $PC2$ je nezmění a všechny sub-klíče budou stejné.

Def (**poloslabé klíče**): dvojice klíčů $(k1, k2)$ taková, že
$E_{k_1}(E_{k_2}(m)) = m \quad \forall m \in M$.
jeden klíč z dvojice dovede dešifrovat zprávu šifrovanou druhým klíčem z páru (protože posloupnost sub-klíčů jednoho z dvojice je opačná jako u druhého).
DES má 6 poloslabých dvojic klíčů:
$01^{14}01^{14}, 10^{14}10^{14}$
$01^{14}10^{14}, 10^{14}01^{14}$
$01^{14}0^{28}, 10^{14}0^{28}$
$01^{14}1^{28}, 10^{14}1^{28}$
$0^{28}01^{14}, 0^{28}10^{14}$
$1^{28}01^{14}, 1^{28}10^{14}$
pozn: $\exists$ i jiné druhy slabosti klíčů ("slabší" něž tyto).

complementation property (komplementarita): $E_{c(k)}(c(m)) = c(E_k(m))$,
kde c značí záměnu 0 za 1 a naopak 1 za 0 (komplement).

doplněk otevřeného textu dá doplněk šifrového, lze využít k luštění:
dvojice šifrových a otevřených textů: $(m, c_1), (c(m), c_2)$ neznáme klíč,
provedeme zašifrování $E_k(m)$,
pokud se výsledek nerovná c_1 můžeme vyloučit k jako správný klíč a navíc:

$$c_2 = E_{c(k)}(c(m)) = c(E_k(m))$$

nejsou-li si krajní výrazy rovny (máme k dispozici!) lze vyloučit i klíč $c(k)$,
chosen-plaintext útok tedy vyžaduje pouze vyzkoušení poloviny klíčového prostoru (tj. v případě DES 2^{55}),
věří se! že to prostě tvůrcům uniklo...

časopaměťovvý útok: Hellman (1980), výpočet všech možných šifrových textů (tedy z jednoho otevřeného přes všechny možné klíče),
pak lze klíč detekovat porovnáním šifrového textu s takto předpřipravenou databází, která je "komprimována" (snížení nároků na paměť, zvýšení nároků na výpočet),
speciální stroj ze 10000 čipů, potřeba asi ročních před-výpočtů, následné hledání klíče do jednoho dne,
nedotažené do konce - spousta otevřených praktických problémů...

1992: návrh programovatelného stroje schopného luštit DES, složený z koherentních procesorů (rozsáhlé jednorozměrné pole jednoduchých paralelních procesorů), obsahově adresovaná paměť, 450 strojů, každý 2^{11} čipů, každý s 2^{10} procesory pak: prostor klíčů lze prohledat za den (náklady v té době 30M$), velký přínos - programovatelnost.

ke konci roku 1992 odhad: 100 čipů za 50.000$ pro luštění 40-bit klíče (tato délka byla povolena jako max mimo USA) pouze ze šifrového textu asi 18min.

Biham, Shamir, konec osmdesátých let, roku 1993 použita k útoku na DES.
snaha zachytit specifické diference ve vstupu vedoucí ke specifickým diferencím ve výstupu, a to takové, které mají podezřele vyšší pravděpodobnost (než by měli mít pokud by šlo o náhodu),
porovnává dvojice otevřených textů a šifrových a hledá dvojice šifrových jejíž otevřené vykazují jisté rozdíly,
vyhledávají se ty které mají vyšší frekvenci výskytu v šifrových - tzv. **charakteristiky**, ty pomohou určit pravděpodobnost klíčů s tím, že výsledkem je nejpravděpodobnější klíč.

použití: několik takových diferencí mající lehce vyšší pravděpodobnosti slouží k odkrytí klíče s využitím chosen-plaintext útoku, příčina S-BOXy a slabé zpracování klíče:
nechť jsou vstupy $R_1 \quad R_2$ a jejich diference $R_1 \; XOR \; R_2$, pak diference nejsou ovlivněny ani expanzí a součtem (modulo 2) s klíčem:
$(E(R_1) \; XOR \; K) \; XOR \; (E(R_2) \; XOR \; K) = E(R_1) \; XOR \; E(R_2) = E(R_1 \; XOR \; R_2)$

nevhodné S-BOXy = pro některé diference vstupů je velká část diferencí výstupů málo pravděpodobné a jiné velmi pravděpodobné.
výstupní diference se v dalším kole stávají vstupními atd... tedy čím více round šifra má tím klesá účinnost metody.

původní metoda ukládala do tabulky možná (pravděpodobná) "doporučovaná" řešení a nakonec se z nich vybral to nejpravděpodobnější (fungovalo celkem dobře na 6-round DES: 240 párů šifrových textů, 0.3s, 8-round: 1500 párů, 2min).

vylepšení: "get-and-try" - správnost doporučeného klíče se okamžitě testuje na známé dvojici otevřeného a šifrového (čas 2^{37} (operací), 2^{36} šifrových textů z 2^{47} vybraných otevřených - nepraktické, nezdá se být realistické předpokládat, že by měl útočník k dispozici tolik informací

pozn: S-BOXy byly zřejmě navrženy aby právě do jisté míry pomohly odolávat tomuto druhu útoku, tvůrci DES prý tuto metodu objevili dříve a proto věděli jak boxy designovat - metodu však nesměli publikovat [říká se..]).

Matsui, 1990+ ('93?)
bere v úvah možnou lineární závislost výstupu na vstupu,
(např. S-BOX $S5$ dává s velkou pravděpodobností lineární vztah mezi některými vstupy a některými výstupy),
metoda nahrazuje nelinearity jejich nejlepšími lineárními aproximacemi.
nepotřebuje vybrané otevřené texty, stačí známé.
přesto vyžaduje velké množství těchto $\longrightarrow$ nepraktické!

nějaké výsledky:
8-round DES: 2^{21} známých otevřených textů $\approx$ 40s,
12-round DES: 2^{33} známých otevřených textů $\approx$ 50h,
pokud lze o otevřeném textu předem něco předpokládat (např. že obsahuje pouze malá písmena) lze luštit přímo ze šifrového textu! (předpoklad anglické texty, pak pro 8-round DES: stačí 2^{29} textů).

od 1988 NSA přestává doporučovat použití DES - problém: rozšíření, kompromis: na dalších 5 let využití mimo federální nefinanční použití, stal se více méně mezinárodním standardem, využití v bankovnictví (např. situace v roce 1993: banky v USA denně prováděly transakce za více než 400 miliard $), také pro šifrování PIN kreditek.

1994: stroj 57600 čipů, cena 1M$, test všech 2^{56} klíčů za 7h (prokazatelně! sestavitelný).

1995: stroje schopné luštit DES do 15min (pro potřeby americké vlády) - prý..

fakta: 56-bit klíč $= 2^{56} =$ 72 057 594 037 927 936 možností,

v době standardizace DES se věří, že stroj schopný prolomit šifru metodou hrubé síly by byl velmi nákladný, pracoval by neúnosně dlouho, byl by značně poruchový...*(a jiné "výmluvy")*.

29.1.1997 - RSA Data Security Inc. (jedna z nejvýznamnějších amerických kryptografických firem) vypisuje soutěž s odměnou 10.000$,
"jednoduchý" úkol: najít klíč metodou hrubé síly.

18.2.1997 - projekt DESCHAL zahájen,
využití velkého množství PC připojených k internetu,
software sepsán skupinou vědců z oblasti informatiky:
Rocke Verser, Justin Dolske, Matt Curtin a kol.
problém: DES nesměl být "vyvážen" mimo území USA a Kanady.
software zkoušel všechny klíče v přidělené části prostoru, umístěn na web - ke stažení zájemcům z USA a Kanady.

13.3.1997 - začalo hledání...
nejprve 20 strojů, později "ve špičce" ke konci až 14.000 strojů,
client software: výkon 1M klíčů/s (maximum na dobovém 200MHz Pentium)
$\rightarrow$ jeden počítač by hledal klíč $\approx$ 2285 let!

17.6.1997, něco před půlnocí - (96 dní po zahájení hledání a 140 po vyhlášení soutěže) je hotovo!

otevřený text: "strong cryptography makes the world a safer place."

klíč: 8558891AB0C851B6

řešení nalezeno na Pentium 90MHz 16MB RAM ($\approx$ 250.000 testů/s), bylo otestováno 17 731 502 968 143 872 možností (tj. 24.6% všech možných) max. výkon: asi 7 miliard testů/s (takto by se řešení našlo za 32 dní), majitel počítače, který nalezl klíč získal odměnu 4.000$.

"úloha by se dala přirovnat k hledání jehly v kupce sena o průměru základny 2.5 míle a výšce 1 míle"

DES CHALLENGE II - další kolo soutěže, odměna jen pokud je luštitelský rekord (doba luštění) překonán aspoň o 25% + deadline 90 dní,
výherce se našel za 40 dní (prohledáno 85% prostoru).

tentokráte se hledalo s pomocí **distributed.net** = distribuované řešení rozsáhlých numerických problémů využívající výkonu málo vytížených CPU/GPU zapojených do systému - asi 50.000 strojů,
max výkon 34 430 460 000 klíčů/s.
otevřený text: "The secret message is: many hands make light work".

DES CHALLENGE II-2 - 17.7.1998 - stroj "Deep Crack" (DES Cracker) dešifruje text za 56 hodin, prokázal praktičnost metody hrubé síly.

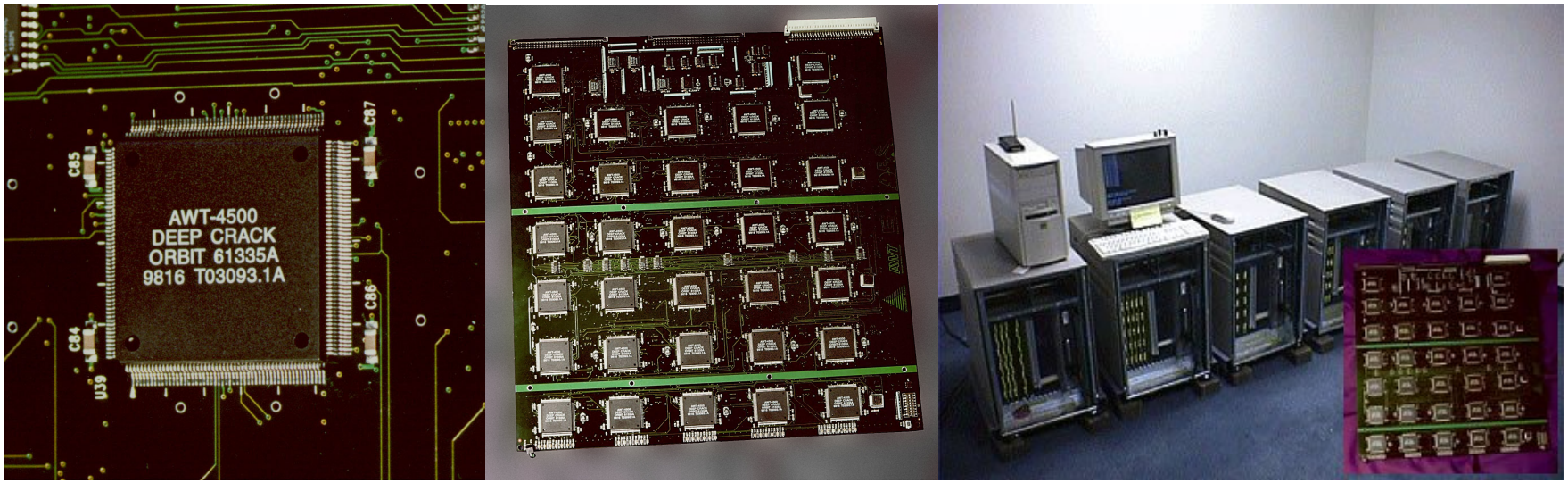
otevřený text: "The secret message is: It's time for those 128-, 192-, and 256-bit keys."

EFF DES Cracker (přezdívaný "Deep Crack") sestavený Electronic Frontier Foundation v roce 1998 (nezisková organizace zabývající se digitálním právem) ve spolupráci s Cryptography Research Inc., Advanced Wireless Technologies, hlavní designer: Paul Kocher (prezident CR).

náklady 250.000$ (!přitom výhra činila 10.000$).
1856 čipů ASIC (**A**pplication **S**pecific **I**ntegrated **C**ircuit), zasazeno do 29 desek (na každé 64 čipů) (sestaveno v AWT), zamontováno do 6 skříní (Sun-4/470) (běžně používané pro servery), vše napojeno na PC.

výkon: test 90 miliard klíčů / s = 9 dnů k otestování všech možných (v průměru stačí polovina této doby).

pozn: proč byl cracker sestaven ze speciálních čipů - (v té době) neekonomické propojit velké množství PC (50.000) když potřebujeme pouze CPU, navíc speciální procesor určený k luštění (DES) efektivnější než obyčejné CPU, počet čipů určuje cenu resp. dobu luštění.



deep crack v celé své kráse (zleva): ASIC čip, deska osázená ASIC čipy, celé zařízení: 6 skříní s deskami a řídící PC.

vyhledávací mini jednotka (24 jich je součástí 1 čipu) obdrží od řídící jednotky (CPU obyčejného PC) 2 bloky šifrového textu a část klíčového prostoru, (horních 24-bitů je pevně dáno, jednotka zkouší kombinace na dolních 32-bitech, začne u samých (32) nul).

jednotka zkusí rozšifrovat první blok - je-li výsledek "nezajímavý" - zkusí další kombinaci bitů jako klíč.

je-li "zajímavý" rozšifruje i druhý - je-li i ten "zajímavý" pošle klíč řídící jednotce a řídící jednotka rozšifruje celý text s tímto klíčem a rozhodne zda jde o řešení,

mezitím mini jednotka pokračuje, když prohledá celý prostor (2^{32}) požádá o nové bloky a část prostoru klíčů.

"zajímavý" výsledek: dán množinou "zajímavých" bajtů a pozicemi, na kterých se mohou vyskytovat (tj. které je třeba kontrolovat),

lze určit na základě nějaké apriorní informace o otevřeném textu (např. víme, že jde o text, který obsahuje pouze velká+malá písmena atd.).

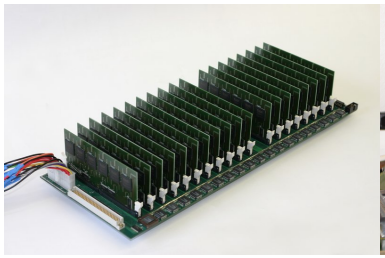
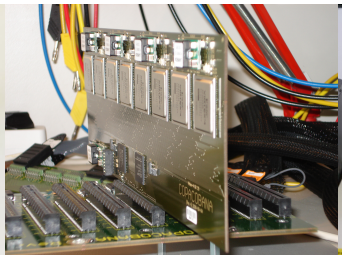
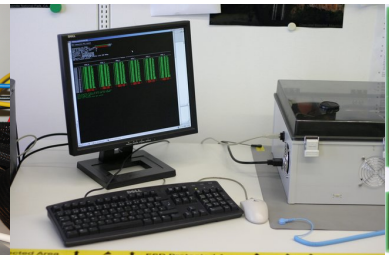
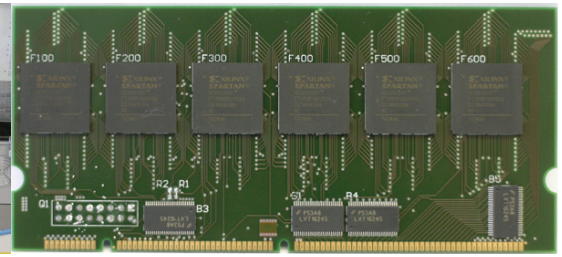
frekvence čipů: 40MHz, mini jednotka - test 2.5M klíčů / s,
tj. čip 60M klíčů /s, deska (64 čipů) skoro 4miliardy klíčů /s,
v průměru stačí prohledat polovinu klíčového prostoru = 110 dní, 29 desek stroje = 4.5 dne

19.1.1999 - Deep Crack ve spolupráci s distributed.net dešifruje za 22h 15m.
"See you in Rome (second AES Conference, March 22-23, 1999)."
důsledky: navržen nový standard Triple-DES (Říjen 1999), výpočetní náročnost a jiné nedokonalosti nakonec vedou k zavedení AES (26.5.2002).

Blaze challenge: najít šifrový text ve formě 8 stejných znaků a klíč, který by tento text dešifroval na otevřený text tvořený také 8 stejnými znaky (obecně jinými než v šifrovém textu),
2.7.1998 - DES Cracker našel řešení: otevřený text = 8x 87,
klíč = 0E 32 92 32 EA 6D 0D 73,
šifrový = samé nuly.

2006 - sestavena **COPACOBANA**, založena na **FPGA** (**F**ield-**P**rogrammable **G**ate **A**rray), COPACOBANA = **CO**st-optimized **PA**rallel **CO**de**B**reaker.
podobný výkon jako "Deep Craker", ale méně nákladný (způsobeno rozvojem v technologiích).
navíc "pracuje" nejen s DES ale i jinými (RC5).

2008 - **COPACOBANA RIVYERA**, vylepšená verze stroje COPACOBANA najde klíč v průměru za den!

DES
jedna z nejdiskutovanějších šifer,

základní forma již nepoužitelná (svého času ovšem nejpoužívanější šifrou), dokonce ani triple varianta (stále ještě někde používán, je bezpečnější ale pomalejší, jinak nový standard - AES),

ovšem - vzhledem k délce klíče (56-bit) překvapivě velmi "bezpečné"!,

nejpraktičtější útok stále hrubá síla (i když $\exists$ teoretické útoky méně náročné na čas ovšem podmínky k jejich realizaci silně! nepraktické).

praktická slabina DESu je krátký klíč - řešení: šifra vycházející z DES, ale používající delší klíč,

modifikovat vnitřní strukturu DES může naopak velmi snadno velmi drasticky snížit kvalitu šifry $\leftarrow$ to není cesta, raději buď vícenásobná aplikace DES nebo úplně něco jiného.

Double-DES

idea zvýšit bezpečnost několikanásobnou aplikací DES (skládání funkcí),
$E_{k_2} \circ E_{k_1}(m) = E_{k_2}(E_{k_1}(m)) = c$
$m = D_{k_1}(D_{k_2}(c)) = D_{k_1} \circ D_{k_2}(c)$

zdálo by se, že bezpečnost se zdvojnásobila (délka klíče se zdvojnásobila),
skutečnost: 56-bit keylength security level (původní DES 55-bit - malé zlepšení).

dk (Merkle, Hellman, 1981) pomocí **meet-in-the-middle** (setkání uprostřed) útoku (Diffie, Hellman, 1977).

pozorování: pokud $E_{k_2}(E_{k_1}(m)) = c$ pak $D_{k_2}(c) = E_{k_1}(m)$
(protože $D_{k_2} \circ E_{k_2}$ je identita).

útok: je dán pár otevřeného a šifrového (m_1, c_1),

sestavíme tabulku T_1 všech možných (seřazených) (2^{56}) hodnot $E_{k_1}(m)$,

sestavujeme další tabulku možných hodnot $D_{k_2}(c)$ a pro každou kontrolujeme hodnoty v T_1:

pokud $\exists$ shoda (označme K_1, K_2) otestujeme shodu na jiném páru (tj. $E_{K_1}(m_2) = D_{K_2}(c_2)$),

pokud se opět shoduje = správné klíče,

náročné na paměť! kterou potřebujeme pro tabulky.

Triple-DES

aka: TDES, 3DES, TDEA (**t**riple **d**ata **e**ncryption **a**lgorithm), trojnásobná aplikace DES na každý blok = zvýšení odolnosti vůči útokům hrubou silou bez nutnosti vytvoření úplně nového algoritmu = pomalejší šifrování.
místo jednoho klíče - balík klíčů (3 DES klíče: k_1, k_2, k_3).

šifrování: $E_{k_3}(D_{k_2}(E_{k_1}(m))) = c$
dešifrování: $D_{k_1}(E_{k_2}(D_{k_3}(c))) = m$
(E, D šifrování resp. dešifrování DES, k_i klíče, c šifrový text, m otevřený)

pozn: pokud $k_1 = k_2$ nebo $k_2 = k_3$ pak: $D_{k_2} \circ E_{k_1}$ nebo $E_{k_2} \circ D_{k_3}$ jsou identity tj. vlastně šifrování obyčejné DESem (resp. pokud dokonce $k_1 = k_2 = k_3$), to je tam právě kvůli zpětné kompatibilitě s DES.
rovněž $\exists$ i varianta trojitého šifrování se dvěma klíči: $E_{k_1}(D_{k_2}(E_{k_1}(m))) = c$

bezpečnost:
3 klíče: opět neplatí, že náročnost útoku by byla 2^{3n} - lze opět aplikovat meet-in-the-middle $\longrightarrow$ snížení na 2^{2n} (i tak lepší než DES i DDES).
2 klíče: hrubou silou 2^{2n}, chosen-plaintext útok (Merkle, Hellman): 2^n ovšem potřebuje 2^n vybraných otevřených textů (nepraktické).

DES-X

Ron Rivest, Květen 1984.
varianta DES vytvořená za účelem znesnadnění útoku metodou hrubé síly (bez nutnosti podstatněji zasahovat do původního algoritmu) přidáním "key whiteningu".

key whitening: kombinování dat s částí klíče pomocí nějaké operace (nejčastěji XOR) před prvním krokem šifrování a po posledním.
první bloková šifra, která tuto metodu použila (později například šifra Twofish).
vyžaduje navíc další dva 64-bitové klíče (pro whitening) - jeden k otevřenému textu před vlastním šifrováním a druhý k aplikaci po zašifrování.

tři klíče k_1, k_2, k_3, šifrování: $k_1 \oplus E_{k_2}(k_3 \oplus m)$,
64-bit klíč k_3 sečten modulo 2 s otevřeným, to je šifrováno s klíčem k_2, a k šifrovému je přičten modulo 2 klíč k_3,
velikost klíče: 56+2x64=184-bitů.

bezpečnost: diferenciální kryptoanalýza: 2^{61} chosen-plaintexts (DES 2^{47}),
lineární kryptoanalýza: 2^{60} (DES 2^{43}).

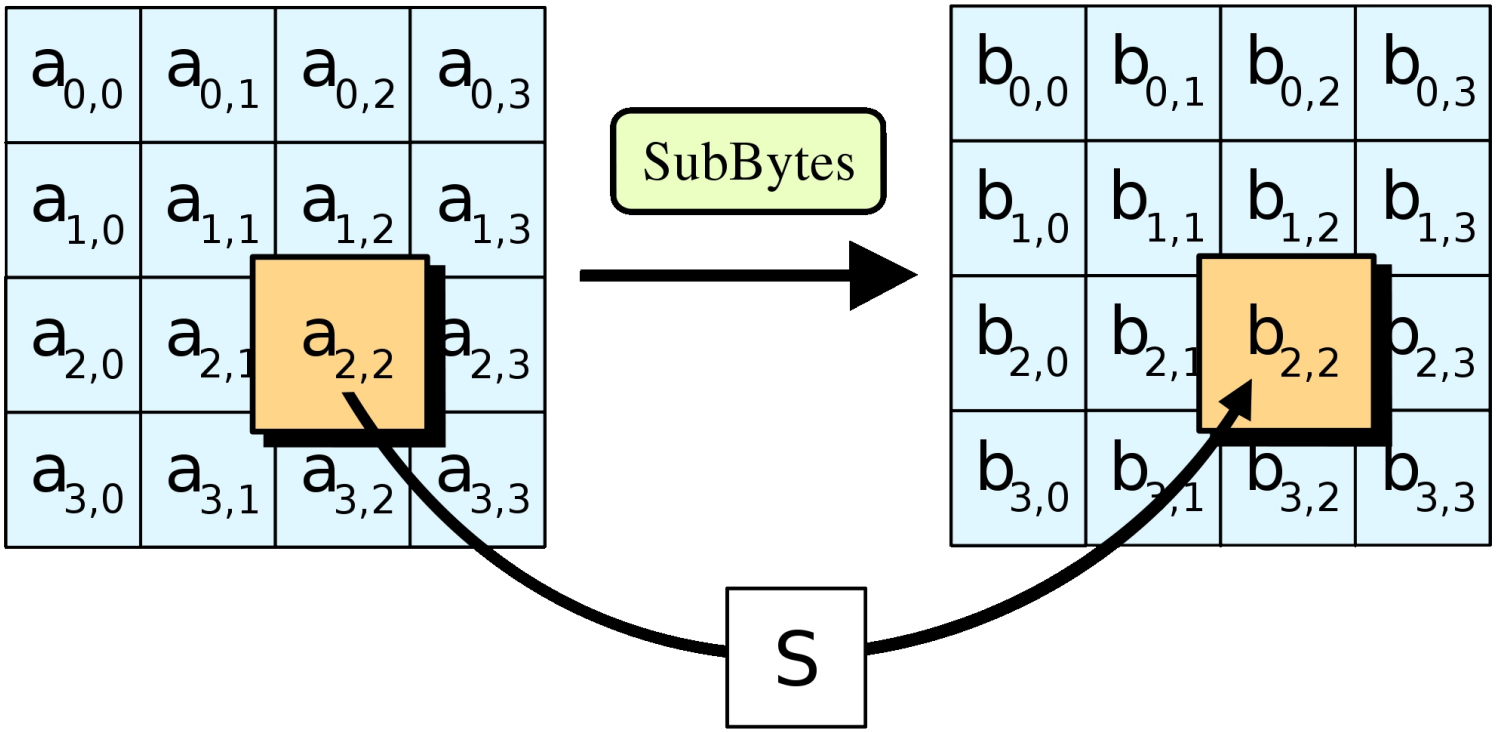
Advanced Encryption Standard Process

1.2.1997 - zahájen proces (první tři měsíce: návrhy na požadavky na nový algoritmus).
12.9.1997 - zahájen výběr kandidátů, požadavky: blokové šifrování, 128-bit bloky, 128, 192, 256-bit klíč ← v té době takových šifer málo (nejznámější: **Square**).
během 9 měsíců vybráno 15 kandidátů.
srpen 1998, březen 1999 - konference **AES1**, **AES2** (kandidáti zkoumáni nejen z hlediska bezpečnosti ale také výkonu a možnosti implementace na různém hardware/software).
srpen 1999 - vybráno pět finalistů: **Rijndael**, **Serpent**, **Twofish**, **RC6**, **MARS**.
duben 2000 - konference **AES3** (mimo jiné s prezentacemi tvůrců).
2.10.2000 - vybrán **Rijndael** (a zahájen proces standardizace).

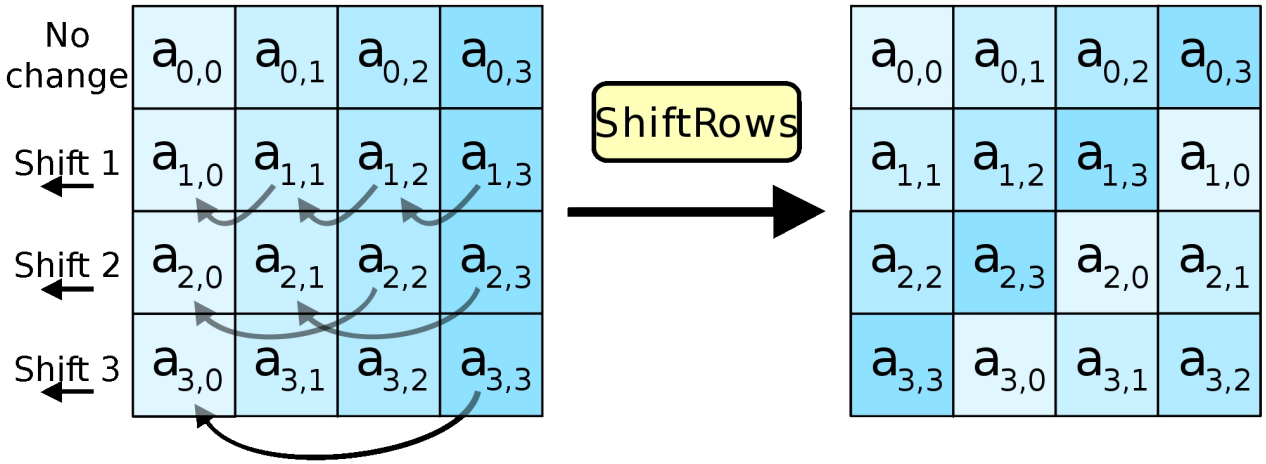
AES (Rijndael)

Vincent Rijmen a Joan Daemen, publikováno 1998.
přijat za standard 26.11.2001.
odvozeno z: **Square**.
AES jako základ dalších šifer: **Anubis**, **Grand Cru** ...
velikost klíče: 128, 192, 256-bitů
velikost bloků: 128-bitů
(původní sada algoritmů **Rijndael**: velikost bloku: násobek-32 (max 256), klíč: násobek-32 (teoreticky neomezeno)).
princip: substitučně permutační (nepoužívá **Feistel síť** na rozdíl od **DES** a jeho zobecnění).
cyklů: 10, 12, 14 (v závislosti na velikosti klíče)
pracuje na 4x4 poli (matici) bajtů (tzv. state).
šifrování: několikerá aplikace transformací, každá se skládá z několika kroků, některé závisí na volbě klíče.
dešifrování = inverzní proces.

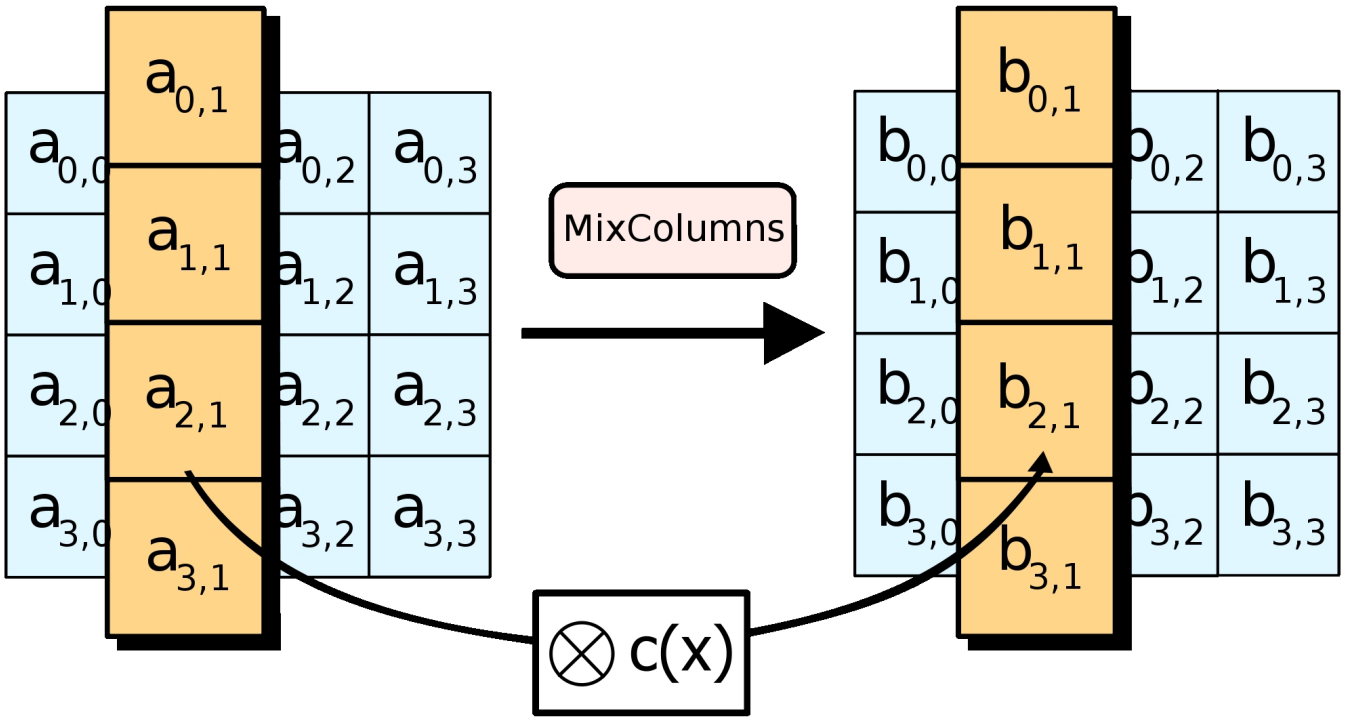
Operace prováděné pouze na začátku:
KeyExpansion - rozklad klíče na několik sub-klíčů pro každou iteraci pomocí několika různých operací (cyklický posun bitů, n-tá mocnina dvou (n = počet iterací), S-BOX).
InitialRound (**AddRoundKey** - každý bajt stavové matice (state) je kombinován se sub-klíčem dané iterace (round key)).
operace prováděné **n**-krát (**n** = počet iterací):
SubBytes - nelineární substituce (každý bajt je nahrazen jiným dle vyhledávací tabulky - **Rijndael S-Box**).
ShiftRows - na každý řádek stavové matice (state) je aplikován cyklický posun.
MixColumns - zkombinování bajtů každého sloupce stavové matice (state).
AddRoundKey
Operace prováděné pouze na konci: stejně jako předchozí ale bez operace **MixColumn**.


a0,0 a0,1 a0,2 a0,3
a1,0 a1,1 a1,2 a1,3
a2,0 a2,1 a2,2 a2,3
a3,0 a3,1 a3,2 a3,3
SubBytes
b0,0 b0,1 b0,2 b0,3
b1,0 b1,1 b1,2 b1,3
b2,0 b2,1 b2,2 b2,3
b3,0 b3,1 b3,2 b3,3
S

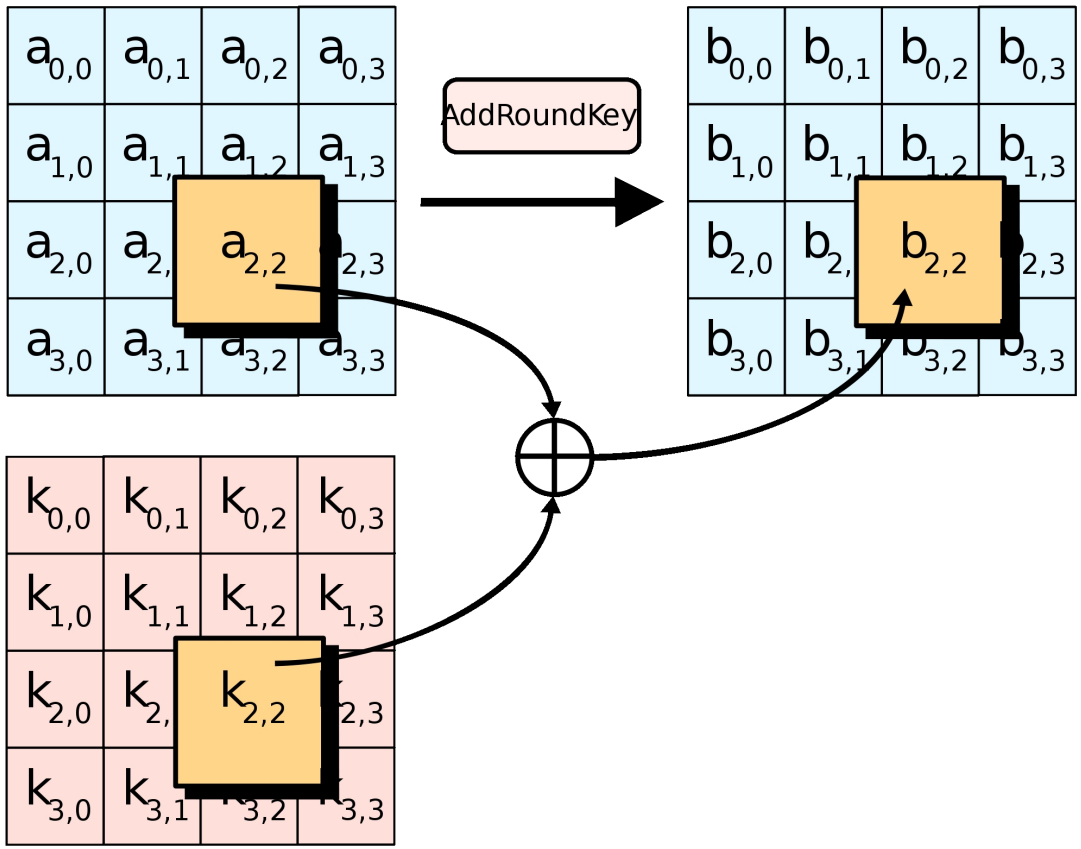
bajty v každém řádku jsou posunuty doleva, v každém řádku je ovšem počet přesouvaných bajtů a délka posunu jiná.



každý sloupec state matice je násoben pevně daným polynomem $c(x) = 3 \times x^3 + x^2 + x + 2$ (spolu s operacemi **ShiftRows** provádí difuzi šifry).



sub-klíč je kombinován s maticí stavu, pro každou iteraci je sub-klíč odvozen ze základního klíče (pomocí Rijndeal key schedule v kroku **KeyExpansion**), jednotlivé bajty klíče jsou bitově XORovány s odpovídajícími bajty sub-klíče.



květen 2009 - side-channel attack (neútočí na šifrování jako takové, ale na konkrétní implementaci jejíž chybou uniknou data).
2002 - **XSL** attack pro případ 128-bit klíče (2^{100} operací $\times$ hrubá síla: 2^{128}) $\leftarrow$ teoretický útok (Nicolas Courtois, Josef Pieprzyk) který měl za cíl poukázat na slabinu zapříčiněnou jednoduchým popisem systému (později se ukázalo, že nemohl fungovat!).
1.7.2009 related-key attack (192/256-bit verze) = útočník sleduje jak se mění šifrování v závislosti na změnách v klíči, kde klíč není celý znám, ale jsou známy jeho jisté vlastnosti (například nějaké bity klíče jsou stejné).

XSL útok

Nicolas Courtois, Josef Pieprzyk, 2002.
založeno na soustavě kvadratických rovnic (pro 128-bit AES 8000 rovnic o 1600 neznámých).
řešení soustavy odkrývá klíč.
výhoda: vyžaduje jen několik otevřených textů.
nevýhoda: náročnost.
na **AES4** konferenci (Bonn, 2004) V. Rijmen okomentoval **XSL**: *"The XSL attack is not an attack. It is a dream."*
...načež mu N. Courtois pohotově odpověděl: *"It will become your nightmare."*

Serpent

finalista **AES**, *2.místo*.
Ross Anderson, Eli Biham, Lars Knudsen, publikováno 21.8.1998.
odvozeno z **Square**.
velikost klíče: 128, 192, 256-bitů.
velikost bloku: 128-bitů (4 x 32-bitová slova).
princip: substitučně-permutační síť.
32 cyklů (dle autorů zaručuje bezpečnost již 16 cyklů) + počáteční a konečná permutace.
v každém z 32 cyklů je aplikován 1 z 8 4-bity-na-4-bity S-BOX (32-krát paralelně), (všechny operace lze provádět paralelně na 32 1-bitových řezech, maximální paralelizace) míchání klíče XORem (první a poslední fáze).

Serpent

nepatentováno, volně k dispozici, lze bezplatně používat, bez omezení, možno implementovat do svého vlastního software/hardware řešení.
bezpečnost:
obecně bezpečnější než **AES**,
teoreticky napadnutelný pomocí **XSL**.
věří se, že implementace **XSL** by byla náročnější než metoda hrubé síly!

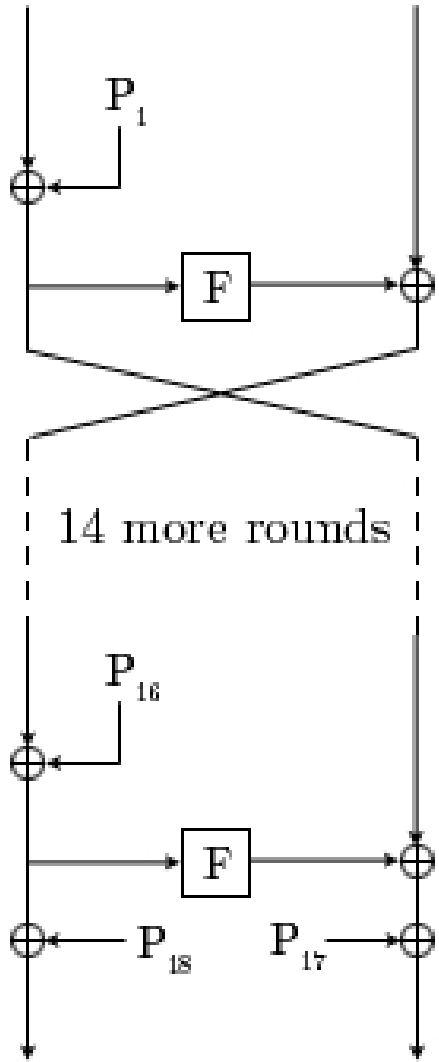
Blowfish

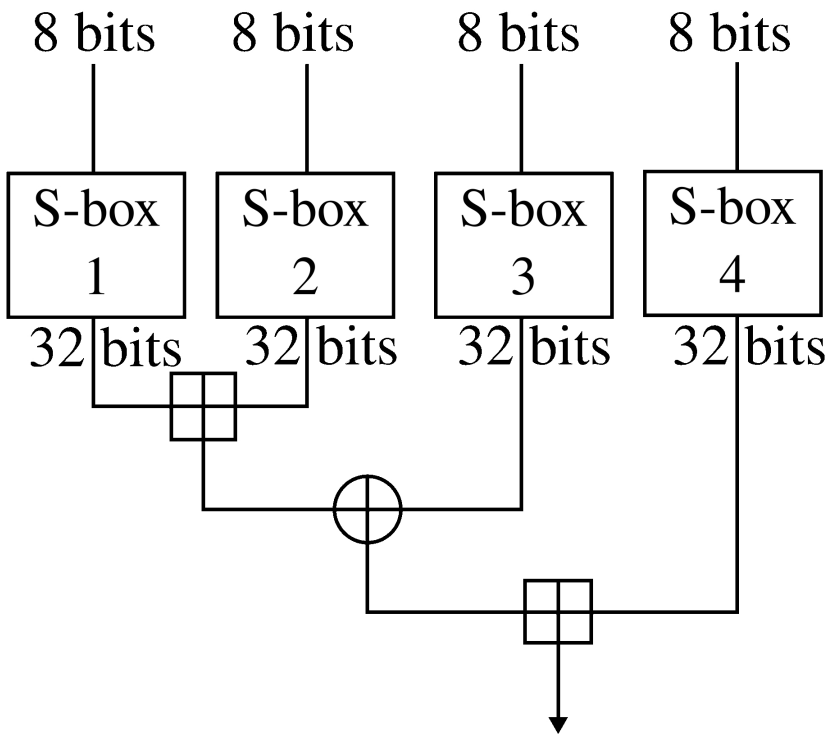
Bruce Schneier, 1993.
velikost klíče: 32-448-bitů v 8-bit krocích (standardně 128-bit klíč).
velikost bloku: 64-bitů
princip: **Feistel síť**
využívá na klíči závislých S-BOXů.
cyklů: 16
efektivní softwareová implementace.
navíc dodnes žádná účinná kryptoanalýza nalezena!
přesto **AES** standard není ani **Blowfish** ani jeho následovník **Twofish**...
vše volně k dispozici (opět žádné patenty).

Blowfish

šifrování

každá řádka diagramu = 32bitů
2 sub-klíčová pole: **P**-array, S-BOX
S-BOX: 8-bit vstup, 32-bit výstup
P-array: 18-ti prvkové pole (jeden vstup použit na začátku, jeden na konci procesu a v každém kroku polovina nepoužitého prvku), prvky XORovány s daty



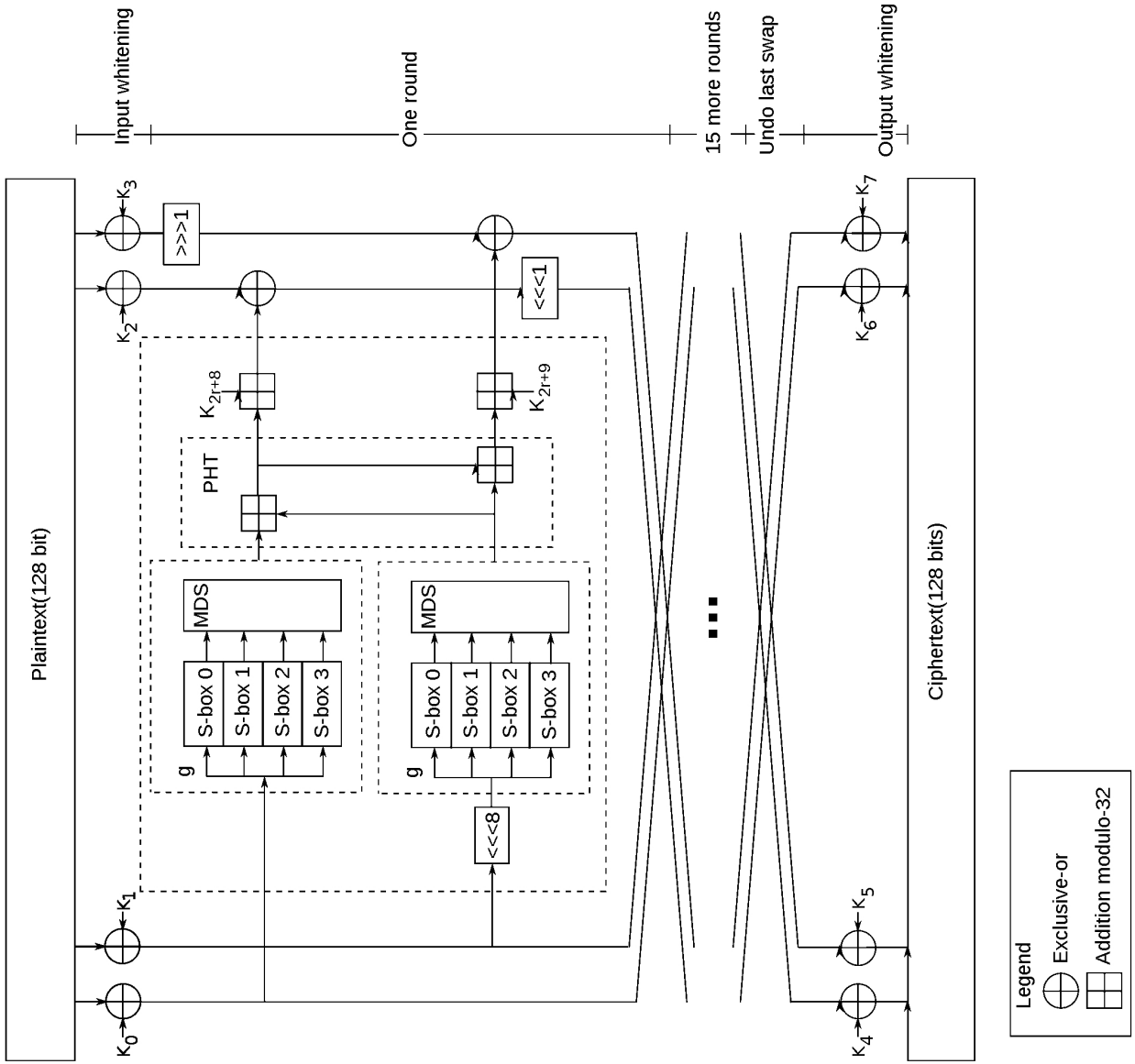


F-funkce rozdělí 32-bit vstup na 4 8-bit části, které slouží jako vstup pro S-BOXy, k výstupům je přičteno modulo 2^{32} a jsou XORovány čímž je získán 32-bit výstup.

Twofish

finalista **AES**, *3. místo*
Bruce Schneier, 1998.
odvozeno z: **Blowfish**, **SAFER**, **Square**
velikost klíče: 128, 192, 256-bitů
velikost bloku: 128-bitů
princip: **Feistel síť** (jako **DES**)
cyklů: 16
předpočítané (na klíči závislé) S-BOXy.
polovina klíče slouží opravdu k šifrování, druhá k modifikaci šifrovacího algoritmu (S-BOXů).

Twofish


Input whitening
One round
15 more rounds
Undo last swap
Output whitening
Plaintext(128 bit)
Ciphertext(128 bits)
K0
K1
K2
K3
K4
K5
K6
K7
>>>1
<<<1
<<<8
K2r+8
K2r+9
PHT
MDS
S-box 0
S-box 1
S-box 2
S-box 3
g
Legend
Exclusive-or
Addition modulo-32

Poznámky ke schématu:
PHT - **P**seudo-**H**adamard transformace (**PHT**) (z šifrovacích algoritmů typu **SAFER**): reversibilní transformace řetězce bitů (sloužící ke kryptografické difuzi), řetěz musí být sudé délky (je rozdělen na dvě stejně dlouhé části). Transformace je pak dána:
$a^{'} = a + b(mod 2^n)$
$b^{'} = a + 2b(mod 2^n)$

MDS - Matice A ($m \times n$) je **MDS** (**M**aximum **D**istance **S**eparable) $\iff$ A je transformační maticí linearní transformace $f(x) : K^n \rightarrow K^m$ (K - konečné těleso) $f(x) = Ax$ taková, že: žádné dvě rozdílné $n+m$-tice $(x, f(x))$ se neliší v n a více prvcích.

literatura

R. A. Mollin: "An Introduction to Cryptography" (Second Edition)
J. Katz, Y. Lindell: "Introduction to Modern Cryptography"
a samozřejmě: http://www.wikipedia.org
a další užitečné odkazy:
http://csrc.nist.gov/archive/aes/
http://gilchrist.ca/jeff/distrib-des2-2.html
http://www.cl.cam.ac.uk/ rja14/serpent.html
http://www.copacobana.org/
http://www.cryptography.com
http://www.cryptosystem.net/aes/
http://www.eff.org
http://www.interhack.net/projects/deschall/
http://www.quadibloc.com/crypto/co4514.htm
http://www.samiam.org/key-schedule.html
http://www.schneier.com/blowfish-products.html
http://www.schneier.com/twofish.html
http://www.usdsi.com/aes.html