Univerzita Karlova, Matematicko-fyzikální fakulta
Ke Karlovu 3, 121 16 Praha 2

Data Encryption Standard

Skúšková práca

Ukázky aplikací matematiky NMAG166

Letný semester 2016/2017

Viktor Bahýľ
Kruh číslo 55
Matematické metódy informačných technológii

Obsah

Úvod

V tejto práci sa budem snažiť popísať šifrovací algoritmus DES čo je skratka pre „*Data Encryption Standard*“. Jedná sa o špeciálny typ iterovanej blokovej šifry. Na začiatok teda pár slov o takýchto šifrách.

Väčšina blokových šifier sú produktové šifry. Tie väčšinou zahŕňajú radu permutácii a substitučných operácií. Bežne používaný systém je takzvaná „*iterated cipher*“ alebo teda iterovaná šifra. Tá pozostáva z definície rundovacej funkcie a zoznamu kľúčov kde zašifrovanie textu prebehne cez daný počet (Nr) podobných (poväčšine rovnakých) opakovaní.

Nech K je náhodný binárny kľúč nejakej danej dĺžky. K je použité na vytvorenie daného počtu (Nr) kľúčov nazývaných rundovacie kľúče, ktoré označujeme $K^1,...,K^{Nr}$. Tieto kľúče spoločne vytvárajú vyššie spomínaný zoznam kľúčov, ten je teda zostavený z K, použitím fixného a verejného algoritmu.

Rundovacia funkcia, označme ju g, má dva vstupy a to určitý rundovací kľúč (K^r) a druhú časť vstupu tvorí súčasný stav (označme ho w^{r-1}). Nasledujúci stav je teda definovaný predpisom $w^r = g(w^{r-1}, K^r)$. Počiatočný stav, w^0, je definovaný ako vstupný text, x. Zašifrovaný text, y, je definovaný ako posledný stav po vykonaní všetkých Nr opakovaní.

Zašifrovanie teda prebieha nasledovne:

$x \rightarrow w^0$, $g(w^0, K^1) \rightarrow w^1$, $g(w^1, K^2) \rightarrow w^2$, ... , $g(w^{Nr-2}, K^{Nr-1}) \rightarrow w^{Nr-1}$, $g(w^{Nr-1}, K^{Nr}) \rightarrow w^{Nr}$, $w^{Nr} \rightarrow y$.

Aby bolo dešifrovanie možné funkcia g musí byť prostá ak jej druhý vstup je fixný. Inak povedané existuje k nej nejaká inverzná funkcia, nazvime ju g^{-1}, spĺňajúca nasledujúci vzťah $g^{-1}(g(w, y), y) = w$ pre všetky w a y. Potom sa dešifrovanie vykonáva nasledovne:

$y \rightarrow w^{Nr}$, $g^{-1}(w^{Nr}, K^{Nr}) \rightarrow w^{Nr-1}$, $g^{-1}(w^{Nr-1}, K^{Nr-1}) \rightarrow w^{Nr-2}$, ... , $g(w^2, K^2) \rightarrow w^1$, $g(w^1, K^1) \rightarrow w^0$, $w^0 \rightarrow x$.

1 Popis

Teraz už prejdime k samotnému algoritmu DES. Ako bolo povedané jedná sa o špeciálny typ rundovacej funkcie opísanej v Úvode tejto práce a konkrétne ju nazývame „*Feistelová šifra*“. Jej základnú formu si teraz opíšeme za použitia terminológie z úvodu. V tejto šifre je každý stav povedzme u^i rozdelený na dve rovnako veľké časti, teda na presné polovice, označme ich L^i a R^i. Rundovacia funkcia g má teraz predpis: $g(L^{i-1}, R^{i-1}, K^i) = (L^i, R^i)$, kde platí $L^i = R^{i-1}$ a $R^i = L^{i-1}$ XOR $f(R^{i-1}, K^i)$. Môžeme si všimnúť, že funkcia f nemusí spĺňať žiadne ďalšie podmienky prostosti, pretože Festelovskú funkciu je vždy možné obrátiť a zmeny vrátiť nasledovne: $L^{i-1} = R^i$ XOR $f(L^i, K^i)$ a $R^{i-1} = L^i$ (viac v kapitole 5).

DES je Feisterova šifra so šestnástimi opakovaniami s veľkosťou spracovávaného bloku šesťdesiatštyri. Zašifruje bitový reťazec x (dĺžky šesťdesiatštyri) použitím päťdesiatšesť bitového kľúča, K. Výstupom celého algoritmu je šesťdesiatštyri bitov dlhý reťazec y, ktorý reprezentuje zašifrovaný text. Samotnému šifrovaniu predchádza fixná počiatočná permutácia IP, ktorá je aplikovaná na vstupný text. Zapisujeme $IP(x) = L^0R^0$. Na konci, po vykonaní všetkých šestnástich opakovaní je aplikovaná inverzná permutácia k tej počiatočnej značená IP^{-1} na bitový reťazec $R^{16}L^{16}$, získavame tak y. Danú operáciu zapisujeme nasledovne $y = IP^{-1}(R^{16}L^{16})$. Všimnime si, že L^{16} a R^{16} sú vymenené pred aplikovaním IP^{-1}. Aplikácia oboch permutácií nemá žiadny kryptografický význam a pri určovaní úrovne bezpečnosti celého algoritmu DES sú často ignorované.

Jedno opakovanie DES algoritmu je zobrazené na obrázku Príloha A. Každé L^i a R^i je dlhé tridsaťdva bitov. Funkcia f, definovaná ako $f : \{0, 1\}^{32} \times \{0, 1\}^{48} \rightarrow \{0, 1\}^{32}$, ma vstup tvorený tridsaťdva bitov dlhým reťazcom (pravá strana súčasného stavu) a rundovací kľúč konkrétneho opakovania. Zoznam kľúčov $(K^1, \ldots, K^{16})$ pozostáva zo štyridsaťosem bitových rundovacích kľúčov, ktoré sa získajú z päťdesiatšesť bitového kľúča K. Každý rundovací kľúč je permutovaná selekcia z K.

Na obrázku v Príloha B je zobrazená bloková schéma funkcie f. V podstate celá funkcia pozostáva zo substitúcie pomocou takzvaných „S-boxov“, ktorá je nasledovaná fixnou permutáciou značenou P. Pre ukážku si prvý vstupný argument f označme A a druhý zase J. Teda pri výpočte výstupu funkcie $f(A, J)$ sa vykonajú nasledujúce kroky:

1. Najskôr je potrebné rozšíriť A na bitový reťazec o dĺžke štyridsaťosem bitov, to vykonáme pomocou fixnej rozširovacej funkcie E (viď Príloha D). $E(A)$ pozostáva z tridsiatich dvoch bitov z A, ktoré sú permutované určitým spôsobom kde sa presne šestnásť bitov objaví dvakrát.
2. Pokračujeme zrátaním logickej operácie XOR medzi výstupom z $E(A)$ a J. Výsledok označíme napríklad B. Bude pozostávať z ôsmich šesť bitových reťazcov, čo zapíšeme $B = B_1B_2B_3B_4B_5B_6B_7B_8$.
3. Tretím krokom je použitie ôsmich „S-boxov“ označených v poradí $S_1, \ldots, S_8$, pričom každý môžeme popísať nasledovne: $S_i : \{0, 1\}^6 \rightarrow \{0, 1\}^4$. Každý z nich teda mapuje šesť bitov do štyroch a je popísaný maticou o rozmeroch 4×16, ktorej obsahom sú celé čísla od 0 do 15 (viď Príloha C). Označme $B_j = b_1b_2b_3b_4b_5b_6$ potom spočítame $S_j(B_j)$ nasledovným spôsobom: dva bity b_1b_6 binárne reprezentujú číslo riadku r v matici S_j (kde $0 \leq r \leq 3$) a zvyšné štyri bity $b_2b_3b_4b_5$ zase

reprezentujú číslo stĺpca c v matici S_j (kde $0 \leq c \leq 15$). Potom je výstup „S-boxu“ definovaný ako číslo nachádzajúce sa v matici na pozícii (r, c) zapísané pomocou binárnej sústavy ako bitový reťazec o dĺžke štyri. Takto postupne zrátame $C_j = S_j(B_j)$ kde $1 \leq j \leq 8$.

4. Nakoniec vytvoríme bitový reťazec $C = C_1C_2C_3C_4C_5C_6C_7C_8$ dĺžky tridsaťdva bitov. Ten potom permutujeme podľa permutácie P (viď Príloha E). Výsledok tejto permutácie $P(C)$ je výstupom celej funkcie $f(A, J)$.

Za požitia tabuliek v Príloha C si teraz ukážeme ako sa počíta výstup pre niektorý z „S-boxov“. Budeme pracovať napríklad s S_1 a za vstup si zvoľme bitový reťazec 011101. Prvý a posledný bit sú 01 čo je binárne reprezentované číslo 1. Prostredné štyri bity sú 1110 čo je binárne reprezentované číslo 14. Takže výstup nájdeme v matici reprezentujúcej S_1 na pozícii (1, 14) čo po nahliadnutí do Príloha C zistíme, že je to číslo 3. To teraz budeme reprezentovať v binárnej sústave: 0011 a teda 0011 je výstup „S-boxu“ S_1 so vstupom 011101.

Ako je jasne vidieť „S-boxy“ nie sú permutácie pretože počet možných vstupov (64) je väčší ako počet možných výstupov (16). Avšak na jednotlivé riadky je možné hľadieť ako na permutácie celých čísel od 0 do 15. Toto je jedna z podmienok, ktoré musia „S-boxy“ spĺňať aby sa zabránilo určitým typom útokov, o tom pojednávajú nasledujúce kapitoly.

2 Analýza

Keď bol DES navrhovaný ako štandard nezaobišlo sa to bez určitej kritiky a nevôle. Jednou z námietok bola otázka „S-boxov“. Všetky výpočty v DES až na jednu výnimku, práve na spomínané „S-boxy“, sú lineárne, napríklad logická operácia XOR s dvomi výstupmi je rovnaká ako vytváranie XOR-u o dvoch vstupoch a potom zratúvať výstup. Keďže „S-boxy“ sú nelineárna operácia je životne dôležitá pre zabezpečenie. V čase keď bol DES predvádzaný, niekoľko ľudí vyjadrilo obavy, že práve v „S-boxoch“ sa môžu nachádzať zadné dvierka, ktoré by mohli Národnej bezpečnostnej službe Americkej vlády (National Security Agency, skrátene NSA) umožniť jednoducho rozšifrovať každú správu zatiaľ čo by falošne mohli tvrdiť, že DES je bezpečný. Samozrejme je nemožné takéto obvinenia vyvrátiť ale žiadny dôkaz podporujúci túto špekuláciu nebol nikdy predložený.

Neskôr bolo dokonca dokázané, že systém „S-boxov“ bol navrhnutý hlavne kvôli bezpečnosti a to konkrétne aby zabraňoval určitému typu útokov. Keď Biham a Shamir prišli s metódou útoku na šifrovacie systémy známou ako diferenčná kryptografická analýza v skorších deväťdesiatych rokoch minulého storočia, bolo zistené, že účel niektorých nezverejnených kritérií „S-boxov“ bol aby bola táto metóda na DES menej účinná. Diferenčná kryptografická analýza bola v IBM známa už počas vývoja DES-u, ale bola udržovaná ako tajomstvo po dobu skoro dvadsiatich rokov, až pokiaľ ju Biham a Shamir neobjavili úplne nezávisle.

Najviac relevantná kritika na DES sú výhrady voči dĺžke šifrovacieho kľúča, 2^{56} poskytuje veľmi malý bitový priestor možných kľúčov nato aby mohol byť DES skutočne bezpečný. Na porovnanie šifrovací systém Lucifer od firmy IBM, ktorý je predchodcom DES-u používal stodvadsaťosem bitový kľúč. Pôvodný návrh DES-u rátal so šesťdesiatštyri bitovým kľúčom ale ten bol neskôr zmenšený až na päťdesiatšesť bitov. IBM tvrdí, že dôvod na toto zníženie bol kvôli nutnosti zakomponovať do kľúča osem kontrolných bitov, čo znamená, že úložný priestor o veľkosti šesťdesiatštyri bitov mohol obsahovať len kľúč o veľkosti päťdesiatšesť bitov.

Už v sedemdesiatich rokoch minulého storočia sa začalo uvažovať o tom, že by bolo možné postaviť špeciálne zariadenie, ktoré by na základe známeho nezašifrovaného textu vedelo vykonať hľadanie šifrovacieho kľúča hrubou silou. To predpokladáme na základe toho, že vieme šesťdesiatštyri bitový text *x* a jeho zašifrovanú podobu *y*. Každý možný kľúč by bol otestovaný pokiaľ by sa nenašiel ten odpovedajúci (vyhovujúcich ich môže byť viacej). Na začiatku roku 1977 Diffie a Hellman potvrdili, že by malo byť možné zostaviť špeciálny čip, ktorý by prešiel celým priestorom možných kľúčov za menej ako deň. Očakávaná cena pre také zariadenie v tom čase bolo dvadsať miliónov amerických dolárov.

Neskôr Michael Wiener v roku 1993 prišiel s veľmi detailným návrhom zariadenia na hľadanie DES kľúča. Toto zariadenie bolo schopné testovať $5*10^7$ kľúčov za sekundu a v roku 1993 by takáto technológia stála 10,50 amerických dolárov za každý čip. Konštrukcia s takým počtom čipov aby bolo možné nájsť v priemere DES kľúč za jeden a pol dna by stála stotisíc amerických dolárov. Keby sa spojilo do jedného zariadenia desať takých konštrukcií (v cene jedného milióna amerických dolárov) priemerný čas potrebný na nájdenie kľúča by klesol na asi tri a pol hodiny.

Wienerov návrh však nikdy nebol uskutočnený a takéto zariadenie sa nikdy nepostavilo. Firma s názvom Electronic Frontier Foundation v roku 1998 postavila zariadenie v cene dvestopäťdesiattisíc amerických dolárov, ktoré je v súčasnosti známe pod názvom „EFF DES cracker“. Tento počítač bol schopný prehľadať osemdesiatosem miliárd kľúčov každú sekundu. Laboratória RSA chceli dokázať, že dĺžka kľúča DES-u nie je dostatočná a preto usporiadali súťaže s názvom DES Challenges. V júly roku 1998 bol na jednej takejto súťaži (DES Challenge II-2) predvedený aj „EFF DES cracker“ a podarilo sa vďaka nemu nájsť kľúč za päťdesiatšesť hodín čím túto súťaž vyhral. V januári 1999 na „DES Challenge III“ bola hlavná úloha (nájsť DES kľúč) vyriešená týmto zariadením pracujúcim spoločne z celosvetovou sieťou (o veľkosti približne stotisíc počítačov, známu tiež pod názvom distrubuted.net) za dvadsaťdva hodín a pätnásť minút, s rýchlosťou spracovávania až dvestoštyridsaťpäť miliárd kľúčov za sekundu.

Okrem hľadania kľúča hrubou silou sú tu ešte dva dôležité kryptografické útoky a to diferenčná a lineárna kryptografická analýza. O týchto metódach si povieme len pár informácii, hlbšie informácii sa dajú nájsť v rade odbornej literatúre a ja sa v tejto práci zameriavam skôr na priblíženie algoritmu DES a na ukázanie jeho kľúčových vlastností viac ako na kompletnú kryptografickú analýzu. Takže pár slov o lineárnej kryptografickej analýze. V prvom rade treba povedať, že tento útok nemá veľký praktický význam kvôli počtu párov čistého a zašifrovaného textu s rovnakým použitým kľúčom, ktorý je potrebný aby mohol byť vôbec uskutočnený. Keď s týmto spôsobom útoku prišiel Matsui tak na vytvorenie všetkých potrebných párov potreboval až štyridsať dní a následne ďalších desať na to aby našiel konkrétny kľúč. Diferenčná analýza je zložitejšia na aplikáciu práve vďaka skorej spomínaným „S-boxom“ a v tejto práci sa jej nebudem venovať (jej všeobecný opis, bez aplikácie na DES je opísaný v kapitole 2.1), podrobnejšie je opísaná v [1].

2.1 Diferenčná kryptografická analýza

V nasledujúcej kapitole sa budeme venovať stručnému popisu diferenčnej kryptografickej analýze. Na akom princípe je záložný tento útok a v čom vlastne spočíva. V prvom rade je dôležité, že človek vykonávajúci takýto útok musí mať prístup k určitému počtu nezašifrovaných správ a k nim aj tie zašifrované. Základný útok spočíva v hľadaní rozdielov medzi podobnými nezašifrovanými správami (podľa tohto sa metóda aj nazýva, rozdiel- diferencia). Tento rozdiel alebo odchýlka sa môže získavať rôznymi spôsobmi najčastejšie je využitie binárnej operácie XOR. Následne sa rovnako spočíta aj rozdiel medzi príslušnými zašifrovanými správami, pričom dúfame, že objavíme nejaké štatistické vzory v ich distribúcií. Výsledný pár rozdielov sa nazýva diferenciál. Ich štatistické vlastnosti záležia na tom ako „S-boxy“ vyzerajú a ako sú navrhnuté. Tieto diferenciály označujeme Δy a Δx kde $\Delta y = S(X \text{ XOR } \Delta x) \text{ XOR } S(X)$ pre každý „S-box“ S. V bežnom útoku je očakávané, že jeden rozdiel bude častejší ako ostatné, tým sa šifra odlišuje od náhodnej. Zložitejšie varianty tejto metódy umožňujú získať šifrovací kľúč rýchlejšie ako pomocou útoku hrubou silou.

V tej najjednoduchšej forme procesu získavania kľúča pomocou tejto metódy potrebujeme poznať veľké množstvo párov zašifrovaného a nezašifrovaného textu, potom predpokladáme, že diferenciál sa nemení aspoň pre r-1 opakovaní, kde r je celkový počet opakovaní danej šifry. Z toho sa potom snažíme vydedukovať, všetky rundovacie kľúče

vhodné pre dané posledné opakovanie. Pokiaľ sú rundovacie kľúče krátke dá sa toto vylúčenie zlých kľúčov vykonať jednoduchým skúšaním všetkých možností pre dané páry šifrovaného a nešifrovaného textu. Pokiaľ sa niektorý kľúč predpokladá ako vhodný pri najväčšom počte párov uvažuje sa o ňom ako o správnom rundovaciom kľúči.

Pre každú šifru musia byť vstupné rozdiely medzi nezašifrovanými textami veľmi obozretne zvolené aby bol útok úspešný. Bežná metóda je sledovanie tých najvhodnejších rozdielov počas celého procesu šifrovania a na základe toho učiniť rozhodnutie. Keďže je tento typ útoku už známy aj pre širokú verejnosť, je braný ako jeden z nárokov na nové šifry, ktoré sú predpokladané, že budú odolné voči útokom tohto typu.

3 História

Pôvod šifrovacieho algoritmu DES siaha do skorých sedemdesiatych rokov minulého storočia. V roku 1972 po vyhotovení štúdie o potrebách americkej vlády v oblasti počítačovej bezpečnosti, hlavný orgán amerických štandardov vyhodnotil, že vláda potrebuje jeden spoločný štandard kvôli šifrovaniu tajných vládnych informácií. Na základe tohto zistenie sa obrátili na NSA so žiadosťou pre šifru, ktorá by spĺňala požadované kritéria. Žiadny z návrhov NSA sa však neukázal ako vhodný. Druhá žiadosť bola uskutočnená v roku 1974. Tentoraz IBM prišlo s kandidátom, ktorý bol považovaný za prijateľného. Jednalo sa o šifru vyvíjanú v rokoch 1973 a 1974 odvodenú od skorej používaného algoritmu nazývanom Lucifer.

Federálny register v marci roku 1975 publikoval navrhovaný DES. Verejnosť bola vyzvaná k otvorenej diskusii a v nasledujúcich dvoch rokoch sa konalo mnoho stretnutí kde sa preberal navrhovaný štandard. Kritika, ohľadom dĺžky kľúča a záhadných „S-boxov“, ktoré mali byť dôkazom vplyvu NSA pri navrhovaní algoritmu, sa objavila veľmi rýchlo. Podozrenie bolo ako už skorej spomínané, že vplyvom NSA boli vytvorené skryté zadné dvierka a že NSA naschvál oslabovalo celkovú bezpečnosť DES algoritmu. Objavilo sa mnoho oficiálnych vyhlásení, ktoré akýkoľvek vplyv NSA na vývoj algoritmu vyvracalo, tieto vyjadrenia sa objavili aj z radou tímu IBM, pracujúceho na návrhu algoritmu. Naopak neskôr odtajnené záznamy NSA o kryptografickej histórii zase tvrdia, že s IBM veľmi úzko spolupracovalo aby bol algoritmus čo najbezpečnejší a tiež vyplávali na povrch aj informácie o tom, že za zmenšovaním dĺžky kľúča stál tlak práve zo strany NSA, kde sa snažili presadiť zníženie až na štyridsaťosem bitov.

Niektoré z podozrení na skryté slabiny v „S-boxoch“ boli v roku 1990 utíšené nezávislou a verejnou publikáciou Eliho Bihama a Adiho Shamira o diferenčnej kryptografickej analýze (referencia na [1], viac v kapitole 2.1), ktoré obsahovalo všeobecnú metódu na prelomenie blokových šifier. Práve vďaka tomu ako boli použité „S-boxy“ navrhnuté, sú odolnejšie voči tejto metóde útokov ako keby boli zvolene náhodne. Čo implikovalo, že IBM o tejto metóde vedelo už pri vývoji DES-u. To sa neskôr aj potvrdilo, v roku 1994 boli zverejnené niektoré z pôvodných utajovaných kritérií pri navrhovaní „S-boxov“. Podľa niektorých zdrojov IBM túto metódu objavilo v roku 1974 a os NSA dostali požiadavku aby to udržali v tajnosti. Samotné IBM neskôr toto zatajovanie obhajovalo slovami, že táto informácia by mohla keby bola v tom čase sprístupnená širokej verejnosti veľmi ovplyvniť národnú bezpečnosť. Všetky dokumenty obsahujúce citlivé informácie IBM zapečatilo a uložilo v sejfe. Po určitom čase verejnosť uznala, že zásahy NSA pri vývoji spravili DES bezpečnejší.

Aj napriek úvodným kritickým ohlasom bol DES schválený ako federálny štandard v roku 1976 a zverejnený začiatkom roka 1977 pod skratkou FIPS PUB 46, pre použitie na všetky neutajované dáta. Postupne bol ako štandard znova obnovovaný v rokoch 1983, 1988, 1993 a znova aj v roku 1999. Neskôr so zmenami známymi ako „Triple DES“. V roku 2002 bol ako štandard nahradený svojim nástupcom zvaným Advanced Encryption Standard (skrátene AES) čo bolo následkom verejnej súťaže. V roku 2005 bol celý tento štandard v akejkoľvek podobe zrušený. Jediné čo bolo ďalej schválené je používanie „Triple DES“ na citlivé vládne informácie.

4 Praktický príklad

V nasledujúcej kapitole budem predvádzať praktický príklad toho ako funguje algoritmus DES v realite. Pripravme si teda nejakú správu, ktorú budeme šifrovať. Označme ju S ako správa. Zvoľme napríklad S = „ViBa2017“. Jedná sa o osem znakovú správu tú keď preložíme do binárnej sústavy pomocou ASCII tabuľky tak dostávame: S = 01010110 01101001 01000010 01100001 00110010 00110000 00110001 00110111. Zvoľme teraz ešte jeden náhodný šesťdesiatštyri bitový kľúč, s takým sa na vstupe DES-u ráta, niektoré bity však nie sú využívané, slúžia na kontrolu, my ju vykonávať nebudeme ale chceme si ukázať celý proces algoritmu. Označme kľúč K a položme K = 01111111 10111101 01110110 10001110 10000011 10000101 00010001 00010111.

Prvým krokom bude získať zoznam rundovacích kľúčov zo zadaného kľúča. Najskôr je potrebné ho permutovať podľa fixnej permutácie PC-1 (viď Príloha F), ktorá vráti len päťdesiatšesť bitov. Dostávame teda K’ = 00111010 00000101 00000111 11001001 11011010 11110000 10110111. Ďalej takto získaný kľúč rozdelíme na dve polovice a získame L_0 a P_0 s konkrétnymi polovicami. Následne sa prevedie 16-krát rovnaká operácia a to taká že sa vždy vezme ľavá časť z predchádzajúcej a posunie sa každý jej bit o daný počet prvkov doľava získaný z tabuľky počtu posunutí (viď Príloha G) a takto vytvoríme spolu šestnásť rôznych ľavých strán a rovnako postupujeme aj pri vytváraní pravých strán. Pre ukážku L_1 = 01110100 00001010 00001111 1000 a P_1 = 00111011 01011110 00010110 1111. Teraz vytvoríme už konkrétne štyridsaťosem bitov dlhé kľúče pre zoznam rundovacích kľúčov a to tak, že spojíme každú získanú ľavú stranu s pravou a všetky tieto kľúče potom permutujeme podľa permutačnej tabuľky PC-2 (viď Príloha H). Napríklad teda dostaneme K_1 = 00010010 11101001 00000011 10100101 10110001 01011101. Zvyšné kľúče dostaneme rovnakým spôsobom.

Keď máme pripravené rundovacie kľúče môžeme sa pustiť do samotnej práce so správou S. Tú sme si už na začiatku previedli do binárnej sústavy a mohli sme si všimnúť, že má veľkosť presne šesťdesiatštyri bitov, čo nie je náhoda, zámerne som správu zvolil v takomto tvare, pretože DES spracováva text po blokoch takejto veľkosti, pokiaľ by to bolo menej jednoducho doplníme toľko bitov aby to vyšlo, najčastejšie sa to rieši doplnením binárnych kódov pre medzery v ASCII tabuľke. Pokiaľ je to viac jednoducho sa správa spracúva na viacej razy. Takže ako prvé na S aplikujeme počiatočnú permutáciu, tiež označovanú IP (viď Príloha I) a dostávame S’ = 00001111 11110001 10000001 11001010 00000000 11111010 00000010 10010101. Následne tento kľúč rozdelíme na dve polovice, ktoré môžeme označiť LT_0 a PT_0. Teraz vykonáme na tieto polovice šestnásť opakovaní, pričom jedno má vstup predchádzajúce polovice a vracia nové polovice, ktoré získa z predchádzajúcich po úpravách zobrazených v Príloha A. Nová ľavá strana je predchádzajúca pravá. Pravú získavame zložitejším spôsobom ktorému sa budeme venovať v nasledujúcom odstavci.

Predpokladajme prvé opakovanie a vstupom nech sú LT_0 a PT_0. LT_1 získame jednoducho, iba priradíme $LT_1 = PT_0$. Funkcia f bola popísaná v kapitole 1 a jej bloková schéma sa nachádza v Príloha B. Teraz si predvedieme len praktický príklad. Ako prvú teda aplikujeme rozširovaciu funkciu E na predchádzajúcu pravú stranu a dostávame $E(PT_0)$ = 10000000 00010111 11110100 00000000 01010100 10101010, vďaka čomu môžeme

vykonať logickú operáciu XOR medzi takto upravenou pravou stranou a prvým rundovacím kľúčom v zozname, teda K_1. Tým získame 10010010 11111110 11110111 10100101 11100101 11110111 čím sa ale vykonávanie funkcie f nekončí pretože my potrebujeme vrátiť len tridsaťdva bitov a v súčasnosti máme štyridsaťosem, tie si rozdelíme na osem rovnakých častí po šesť bitov a každú ideme spracovávať s odpovedajúcim „S-boxom“, prvá šestica je 100100. Prvý a posledný bit tvoria v desiatkovej sústave číslo 2, prostredné štyri tiež číslo 2 a teda sa pozrieme do tabuľky požadovaného „S-boxu“ (viď Príloha C) a vidíme, že návratová hodnota je číslo 14, zapísané v binárnej sústave je to 1110 a takto spracujeme všetky šestice s konkrétnou tabuľkou „S-boxu“ a ako výsledok dostaneme vždy štvoricu bitov, ktoré dokopy vytvoria: 11100010 01011011 00011011 11000000, čo následne permutujeme pomocou permutácie P (viď Príloha E) a tým získavame návrat funkcie f: 11110100 11110001 10100010 01000001.

Návrat funkcie f ešte spojíme s pôvodnou ľavou stranou pomocou logickej operácie XOR a zapíšeme to do novej pravej strany, teda PT_1 = 11111011 00000000 00100011 10001011. Takto postupujeme ďalších šestnásťkrát a po poslednom opakovaní prehodíme ešte raz získane strany, následne ich spojíme a vykonáme inverznú permutáciu k tej počiatočnej, značenú IP^{-1} (viď Príloha J). V našom príklade dostávame po vykonaní všetkých iterácii aj poslednom prehodení strán nasledujúci reťazec bitov: 11110011 10011111 01000010 00011001 00101100 01011001 01011100 00010000 a po vykonaní permutácie získavame už samotný zašifrovaný text a teda tvar našej zašifrovanej správy, označme ju C = 01110001 01010100 10011000 10111001 01111011 11000000 01101100 01010000. Takto získané bity by sme ešte mohli podľa ASCII tabuľky previesť na znaky ale v mnohých prípadoch sa stáva, že dané kódy odpovedajú znakom, ktoré sa bežne nezobrazujú a teda táto operácia nemá veľký význam. Čo sa týka dešifrovania, vykonáva sa rovnako ako šifrovanie len sa zmení poradie aplikovania kľúčov napríklad pri prvom opakovaní sa použije posledný (šestnásty) kľúč. Je to zabezpečené samotným návrhom algoritmu a tým aké boli na DES kladené požiadavky pri jeho návrhu, viac v kapitole 5.

5 Dešifrovanie

Z predchádzajúceho textu nie je úplne jasné prečo na proces dešifrovania stačí vykonať ten istý algoritmus len s pozmeneným poradím aplikovania jednotlivých rundovacích kľúčov, preto si to v nasledujúcej kapitole priblížime a ukážeme.

Dešifrovanie je vlastne vrátenie všetkých zmien vykonaných na danom reťazci bitov. Počiatočná permutácia IP a konečná permutácia IP^{-1} sú navzájom inverzné a teda vykonaním IP na zašifrovanú správu vrátime posledný krok šifrovania a pokiaľ predpokladáme, že po vykonaní šestnástich opakovaní rundovnej funkcie a následným otočením strán dostaneme rovnaký reťazec bitov aký sme mali v procese šifrovania po aplikovaní IP, túto zmenu vrátime jednoducho vykonaním permutácie IP^{-1} a tým pádom dostávame pôvodný reťazec bitov, z ktorého už pomocou ASCII tabuľky jednoducho dostaneme pôvodný text.

Pozrime sa teraz na jednotlivé opakovania rundovacej funkcie. V poslednom opakovaní pri šifrovaní máme vstup označený ako $L_{15}P_{15}$ kde teda výstup ako bolo povedané je tvorený tak, že pravá strana (P_{15}) sa prehodí na ľavú čím vznikne L_{16}. Ľavú stranu (L_{15}) upravíme pomocou funkcie XOR spojením s výsledkom funkcie *f* s použitým rundovacím kľúčom K_{16} a s reťazcom P_{15} použitým ako vstup do funkcie (ako tieto úpravy prebiehajú bolo už opísané v kapitole 1 a na príklade v kapitole 4), tým dostaneme P_{16}. Teda výsledok všetkých opakovaní je $L_{16}P_{16}$ čo sa ešte pred vykonaním IP^{-1} otočilo číže v procese dešifrovania nám vstup do jednotlivých opakovaní tvorí reťazec v tvare $P_{16}L_{16}$. Na ktorý teraz vykonáme jedno opakovanie rundovacej funkcie. Novú ľavú stranu bude tvoriť pravá polovica vstupu, čo keď si uvedomíme je vlastne L_{16} a to je presne P_{15}. Novú pravú stranu dostaneme spojením P_{16} a výsledkom funkcie *f* s použitým kľúčom K_{16} (pretože pri dešifrovaní sa otáča poradie aplikovania kľúčov) pomocou funkcie XOR. Vstup do funkcie *f* tvorí reťazec L_{16} čo je vlastne rovnaký reťazec ako P_{15} a keďže sa používa rovnaký kľúč ako pri šifrovaní a aj rovnaký vstup, musí byť aj výsledok rovnaký. Teraz vieme, že platí $P_{16} = L_{15}$ XOR $f(P_{15}, K_{16})$, ďalej, že $L_{16} = P_{15}$ a chceme ukázať, že $L_{15} = P_{16}$ XOR $f(L_{16}, K_{16})$. Keďže $L_{16} = P_{15}$ potom platí aj $f(L_{16}, K_{16}) = f(P_{15}, K_{16})$ a jedna zo základných vlastností funkcie XOR (viac o vlastnostiach funkcie XOR v kapitole 5.1) je vlastná inverzia, teda pre každý reťazec A platí A XOR A = 0, kde 0 je reťazec pozostávajúci len z núl, teda ak pre všetky reťazce X, Y a Z platí X = Y XOR Z, úpravami dostaneme X XOR Z = Y XOR Z XOR Z a z u vlastností funkcie XOR teda dostaneme X XOR Z = Y. Z tohto všetkého teda vyplýva, že ak platí $P_{16} = L_{15}$ XOR $f(P_{15}, K_{16})$ musí platiť aj $L_{15} = P_{16}$ XOR $f(L_{16}, K_{16})$.

Podobnú úvahu môžeme vykonať pre každé jednotlivé opakovanie rundovacej funkcie pri šifrovaní a odpovedajúcemu opakovaniu v algoritme dešifrovania. Keďže do prvého opakovania šifrovania bol vstup tvorený reťazcom povedzme L_0P_0 po vykonaní šestnásteho opakovania pri dešifrovaní dostávame ako výsledok reťazec v tvare P_0L_0 čo po vykonaní posledného otočenia bude rovnaký reťazec aký vstupoval do prvého opakovania rundovacej funkcie algoritmu šifrovania.

Teda z vyššie opísaného je vidieť, že vykonanie rovnakého algoritmu len so zmenou v poradí aplikovania kľúčov je presne proces potrebný na vrátenie všetkých vykonaných zmien šifrovania a teda sa jedná aj o algoritmus dešifrovania, ktorý môžeme použiť za predpokladu, že poznáme jednotlivé kľúče.

5.1 Vlastnosti funkcie XOR

Funkcia XOR (alebo aj binárna operácia XOR) má pre všetky bitové reťazce A, B a C dĺžky N, kde N je prirodzené číslo, nasledujúce vlastnosti:

1. Komutatívnosť: platí A XOR B = B XOR A,
2. Asociatívnosť: platí (A XOR B) XOR C = A XOR (B XOR C),
3. Existuje bitový reťazec dĺžky N, označovaný 0, pre ktorý platí A XOR 0 = A,
4. Každý prvok je sám sebe inverzný: platí A XOR A = 0.

Záver

Na záver mojej práce by som rád zhodnotil všetko čo som sa dozvedel aj čo bolo nakoniec jej obsahom. Pri hľadaní vhodných materiálov som si prešiel cez veľké množstvo kníh, článkov, prác a internetových stránok zameriavajúcich sa na tému algoritmu DES. Musím povedať, že som sa o ňom veľa dozvedel a aj keď je to už nepoužívaný a ako vieme nie bezpečný šifrovací algoritmus veľa som sa vďaka jeho štúdií o kryptografii dozvedel. Či už to bolo z jeho histórie, ktorá mi dovolila nahliadnuť ako takéto algoritmy dakedy vznikali, čo všetko a aká politika za tým stála. Ďalej som v tejto práci aspoň trochu priblížil aké má DES slabiny a prečo sa prestal používať, zameral som sa však hlavne na samotný algoritmus a jeho opis, ktorý bol predvedený aj na praktickom príklade. Okrem toho som aj rozobral algoritmus dešifrovania a vysvetlil prečo je až na poradie aplikovania rundovacích kľúčov rovnaký aký sa používa na zašifrovanie.

Z vlastnej vôle, chuti a inšpirácie získanej pri hľadaní informácii som sa rozhodol aj napísať si vlastný program, ktorý slúži na šifrovanie správ a používa tento algoritmus. Na implementáciu do kódu bol tento algoritmus prekvapivo jednoduchý a veľmi často ho využívam na rôzne účely. Použil som ho aj pri spracovávaní materiálov ohľadom kapitoly obsahujúcej praktický príklad, kde som si výsledky overoval priamo použitím tohto programu.

Som veľmi spokojný s výberom témy a rozhodne sa mi svet kryptografie priblížil vďaka získavaniu potrebných informácii o niečo viac a myslím si, že v súčasnom svete je to neutíchajúca téma, vhodná na mnohé diskusie. Táto práca pre mňa slúži aj ako motivácia pre ďalšie štúdium v danej problematike a získavaní stále novších informácií.

Zoznam použitej literatúry

[1] E. Biham a A. Shamir, *Differential Cryptoanalysis of DES-like Cryptosystems,* 1990.

[2] Wikipédia, „en.wikipedia.org," [Online]. Available: https://en.wikipedia.org/wiki/Data_Encryption_Standard. [Cit. 20 Júl 2017].

[3] S. L. Pfleeger a C. P. Pfleeger, „flylip.com," [Online]. Available: http://flylib.com/books/en/4.270.1.136/1/. [Cit. 20 Júl 2017].

[4] W. Bassalee a Y. Stein, „embedded.com," 23 Jún 2008. [Online]. Available: http://www.embedded.com/print/4007609. [Cit. 20 Júl 2017].

[5] C. Diorio, „washington.edu," [Online]. Available: https://courses.cs.washington.edu/courses/cse467/99au/admin/Slides/Week6Lecture1/sld020.htm. [Cit. 20 Júl 2017].

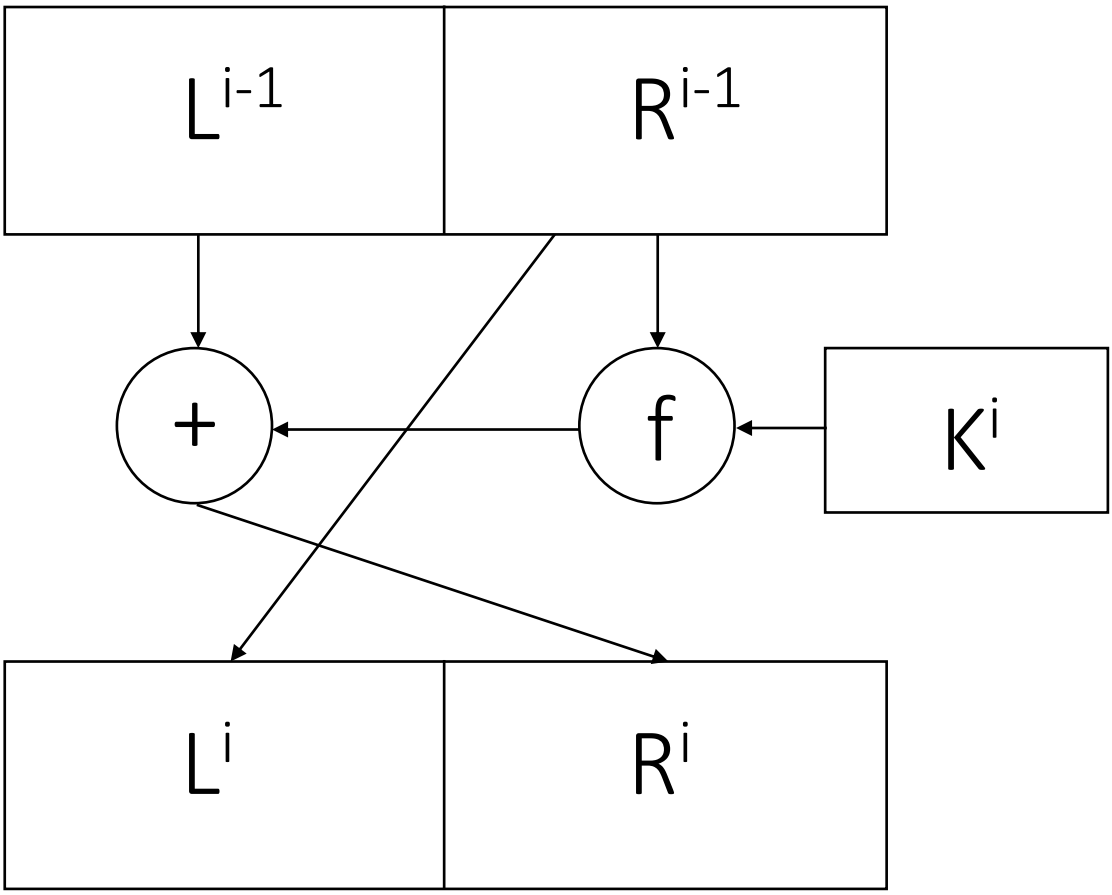
[6] J. Santos a P. Rosa, „paginas.fe.up.pt," 2014. [Online]. Available: http://paginas.fe.up.pt/~ei10109/ca/des.html. [Cit. 20 Júl 2017].

[7] D. R. Stinson, Cryptography - Theory and Practice, Druhý ed., Watrloo: Champman & Hall/CRC, 2002.

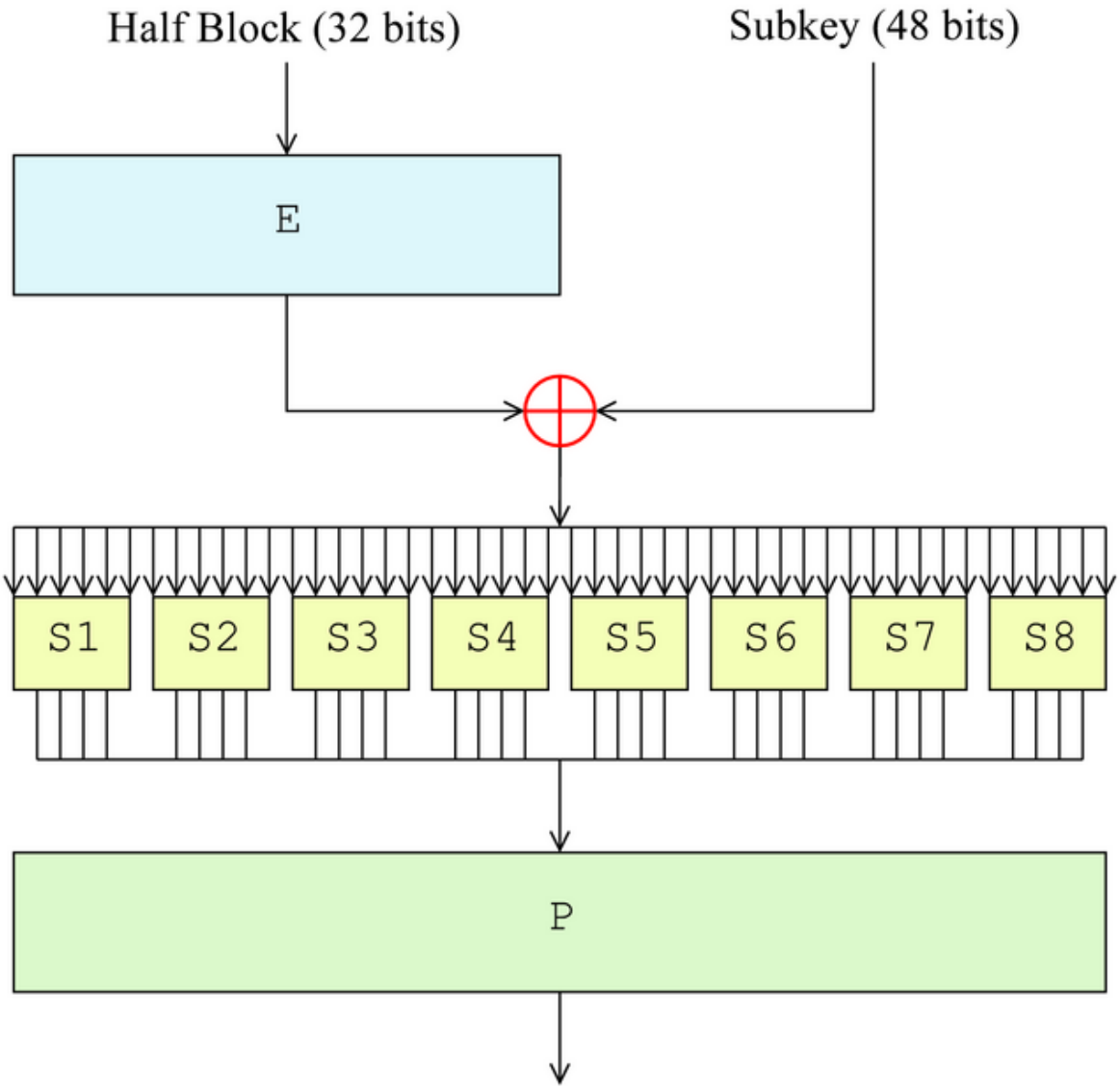
[8] Wikipédia, „en.wikipedia.org," [Online]. Available: https://en.wikipedia.org/wiki/EFF_DES_cracker. [Cit. 20 Júl 2017].

Zoznam príloh

Prílohy



Príloha A - Jedno opakovanie DES šifrovania



Príloha B - Funkcia f [2]

		Column															
Box	Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1																	
	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2																	
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																	
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																	
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5																	
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																	
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																	
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																	
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Príloha C - "S-boxy" [3]

32	**1**	**2**	**3**	**4**	**5**
4	**5**	**6**	**7**	**8**	**9**
8	**9**	**10**	**11**	**12**	**13**
12	**13**	**14**	**15**	**16**	**17**
16	**17**	**18**	**19**	**20**	**21**
20	**21**	**22**	**23**	**24**	**25**
24	**25**	**26**	**27**	**28**	**29**
28	**29**	**30**	**31**	**32**	**1**

Príloha D - Tabuľka rožisrovacej funkcie E [4]

The permutation table

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Príloha E - Tabuľka permutácie P [5]

PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Príloha F - Tabuľka permutácie PC-1 [6]

Iteration Number	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Príloha G - Tabuľka posunutí [6]

PC-2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Príloha H - Tabuľka permutácie PC-2 [6]

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Príloha I - Tabuľka permutácie IP [6]

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Príloha J - Tabuľka permutácie IP^{-1} [6]