Tehnici de CRIPTARE

Claudiu Soroiu

În acest număr continuăm prezentarea tehnicilor de criptare cu descrierea metodelor S-DES și DES standard, tehnicile de criptare cu cea mai largă utilizare.

Simple Data Encryption Standard

Tehnica de criptare ***DES*** (***D**ata **E**ncryption **S**tandard*) a apărut în anii '70 în *Statele Unite ale Americii*. Standardul de criptare a fost realizat de *Biroul Național de Standardizare* (*National Bureau of Standards*) cu ajutorul *Agenției Naționale de Securitate* (*National Security Agency*). Această tehnică de criptare este una dintre cele mai utilizate. Scopul acestei tehnici este acela de a asigura o metodă standard pentru protejarea datelor comerciale neclasificate. *IBM* a creat în anul 1976 o primă implementare a algoritmului sub numele de *Lucifer*.

Acest algoritm utilizează reprezentarea binară a codurilor numerice ale caracterlor folosite pentru reprezentarea datelor care urmează a fi criptate.

Algoritmul de bază *DES* folosește blocuri de 64 de biți de date și aplică doar două operații asupra intrării: deplasare și substituție de biți. Cheia de criptare controlează procesul de codificare a datelor. Prin aplicarea celor două operații în mod repetat și neliniar, se obține un rezultat care, practic, nu poate fi decriptat decât dacă se cunoaște cheia de criptare.

Asupra fiecărui bloc de date, algoritmul se aplică de până la 16 ori (*DES* standard aplică algoritmul de criptare de 16 ori).

Datorită complexității ridicate a algoritmului *DES* standard, în continuare vă prezentăm varianta simplificată a algoritmului *DES*, *Simple-DES*, care folosește blocuri de 8 biți de date.

Algoritmul de criptare

Algoritmul *DES simplificat* poate fi exprimat ca o compunere de cinci funcții:

- o permutare inițială *IP*, care permută biții blocului de intrare;
- o funcție *f*, care depinde de o subcheie k_1 a unei chei de criptare *K*; funcția este foarte complexă și acționează asupra jumătății stângi a blocului de date rezultat la pasul anterior;
- o permutare *SW*, care interschimbă cele două jumătăți ale blocului de date rezultat la pasul anterior. Dacă blocul de date este $d = d_1d_2d_3d_4d_5d_6d_7d_8$, atunci acesta devine $d_5d_6d_7d_8d_1d_2d_3d_4$, unde d_i reprezintă al *i*-lea bit din blocul *d*;
- aceeași funcție *f*, care folosește o altă subcheie (k_2) a cheii de criptare *K*;
- inversa permutării *IP*, IP^{-1}.

Formula matematică ce stă la baza algoritmului de criptare este:

$$Y = E_k(X) = IP^{-1}(f_{k_2}(SW(f_{k_1}(IP(X))))),$$

unde E_K este funcția de criptare care folosește cheia de criptare *K* și se aplică asupra blocului de intrare *X*, rezultând blocul criptat *Y*; f_{k_1} și f_{k_2} reprezintă funcția *f* aplicată asupra unui bloc de date, folosind subcheile k_1, respectiv k_2.

În cazul acestui algoritm fiecare bloc de intrare are 8 biți și cheia de criptare, care este partajată între emițător și receptor, are lungimea de 10 biți. Cheia de criptare este folosită pentru a crea două subchei având fiecare 8 biți.

La început se construiesc cele două subchei de criptare k_1 și k_2. Algoritmul este prezentat în cele ce urmează.

Considerăm cheia de criptare $b = b_1b_2b_3b_4b_5b_6b_7b_8b_9b_{10}$.

Asupra cheii de criptare *b*, se aplică o permutare și se obține un bloc *c* care are 10 biți.

$$c = c_1c_2c_3c_4c_5c_6c_7c_8c_9c_{10} = b_3b_5b_2b_7b_4b_{10}b_1b_9b_8b_6.$$

Cele două jumătăți ale blocului *c* sunt deplasate circular la stânga cu 1 bit și componentele rezultate sunt folosite pentru a construi un bloc *d* care are, de asemenea, 10 biți:

$$c_1c_2c_3c_4c_5 \rightarrow c_2c_3c_4c_5c_1$$
$$c_6c_7c_8c_9c_{10} \rightarrow c_7c_8c_9c_{10}c_6$$
$$c_2c_3c_4c_5c_1c_7c_8c_9c_{10}c_6 = d$$

La pasul următor, ultimii 8 biți ai blocului *d* sunt permutați și se obține prima subcheie, k_1.

$$d_3d_4d_5d_6d_7d_8d_9d_{10} \rightarrow d_6d_3d_7d_4d_8d_5d_{10}d_9 = k_1$$

În continuare, cele două jumătăți ale blocului *d* sunt deplasate circular la stânga cu două poziții, și rezultatul este folosit pentru a obține un bloc de date *e*, care are 10 biți.

$$d_1d_2d_3d_4d_5 \rightarrow d_3d_4d_5d_1d_2$$
$$d_6d_7d_8d_9d_{10} \rightarrow d_8d_9d_{10}d_6d_7$$
$$d_3d_4d_5d_1d_2d_8d_9d_{10}d_6d_7 = e$$

Ultimii 8 biți ai blocului *e* sunt permutați folosind aceeași permutare (cea utilizată și pentru ultimii 8 biți ai blocului *d*), iar rezulatul obținut reprezintă cea de-a doua subcheie k_2.

$$e_3e_4e_5e_6e_7e_8e_9e_{10} \rightarrow e_6e_3e_7e_4e_8e_5e_{10}e_9 = k_2$$

Pentru a construi funcția de criptare *f*, considerăm că blocul de intrare de 8 biți, folosit de aceasta, este format din două părți (stângă - *L* și dreaptă - *R*) de câte patru biți fiecare. Cu această condiție, funcția *f* este:

$$f_k(L, R) = (L \oplus F(R, k), R),$$

unde $\oplus$ reprezintă operatorul binar *SAU-exclusiv*, și $L \oplus F(R, k)$ împreună cu *R* formează blocul de ieșire de 8 biți. *k* reprezintă o subcheie de criptare de 8 biți. Funcția *F* primește ca parametri un bloc de date de 4 biți și o subcheie *k*, și furnizează la ieșire un bloc de 4 biți.

Se observă că funcția *f* modifică numai partea stângă a blocului de intrare, urmând ca după aplicarea funcției *SW* să se modifice și cealaltă jumătate a blocului.

În continuare vom descrie funcția *F*.

Fie $R = n_1n_2n_3n_4$. Asupra lui *R* se aplică două permutări și rezultatele celor două permutări formează un bloc de 8 biți, această operație purtând numele de *expandare*.

$$n_1n_2n_3n_4 \rightarrow n_4n_1n_2n_3$$
$$n_1n_2n_3n_4 \rightarrow n_2n_3n_4n_1$$
$$n_4n_1n_2n_3, n_2n_3n_4n_1 \rightarrow n_4n_1n_2n_3n_2n_3n_4n_1$$

Blocul de 8 biți se va scrie astfel:

$$n_4 \mid n_1 \;\; n_2 \mid n_3$$
$$n_2 \mid n_3 \;\; n_4 \mid n_1$$

La pasul următor se aplică, asupra acestui bloc, cheia de criptare *k* astfel:

$$n_4 \oplus k_1 \mid n_1 \oplus k_2 \;\; n_2 \oplus k_3 \mid n_3 \oplus k_4 \rightarrow p_{00} \mid p_{01} \;\; p_{02} \mid p_{03}$$
$$n_2 \oplus k_5 \mid n_3 \oplus k_6 \;\; n_4 \oplus k_7 \mid n_1 \oplus k_8 \rightarrow p_{10} \mid p_{11} \;\; p_{12} \mid p_{13}$$

În continuare se definesc două matrice S_1 și S_2 numite și ***S-boxes*** (*substitution boxes* - matrice de substituție), astfel:

S_0 =

	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	0	2

S_1 =

	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	2
3	2	1	0	3

Cele două matrice de substituție se folosesc astfel:

- biții p_{00} și p_{03} formează un număr cuprins între 0 și 3 care reprezintă numărul unei linii l_0 din matricea S_0;
- biții p_{10} și p_{13} formează un număr cuprins între 0 și 3 care reprezintă numărul unei linii l_1 din matricea S_1;
- biții p_{01} și p_{02} formează un număr cuprins între 0 și 3 care reprezintă numărul unei coloane c_0 din matricea S_0;
- biții p_{11} și p_{12} formează un număr cuprins între 0 și 3 care reprezintă numărul unei coloane c_1 din matricea S_1;
- numărul de pe poziția l_0, c_0 din matricea S_0 se scrie în baza 2 și se obțin 2 biți m_1 și m_2;
- numărul de pe poziția l_1, c_1 din matricea S_1 se scrie în baza 2 și se obțin 2 biți m_3 și m_4;
- în final se obține blocul de 4 biți $m_1m_2m_3m_4$ căruia i se aplică o permutare și devine $m_2m_4m_3m_1$.

Algoritmul de decriptare este dat de inversa funcției E_k.

Datorită faptului că algoritmul *S-DES* folosește chei de criptare care au lungimea de 10 biți, există doar 1024 posibile chei de criptare. Așadar, un text criptat folosind această metodă poate fi decriptat, în eventualitatea în care nu se cunosc cheile de criptare, folosind un atac cu text necriptat și fiecare cheie de criptare posibilă sau dicționare de chei sau de parole.

Mai mult de 20 de ani s-a crezut că algoritmii de criptare *DES standard* și alte variante ale acestuia sunt atât de siguri încât mesajele criptate cu acesta nu pot fi decriptate decât dacă se cunosc cheile de criptare.

Algoritmul de criptare DES complet

Algoritmul *DES* standard folosește ca intrare blocuri de 64 de biți și chei de criptare de 56 de biți (ceea ce înseamnă că există 2^{56} = 72.057.594.037.927.936 de chei posibile distincte). La baza acestui algoritm se află următoarea formulă matematică:

$$Y = IP^{-1} \circ f_{k_{16}} \circ SW \circ f_{k_{15}} \circ SW \circ ... \circ SW \circ f_{k_1} \circ IP(X)$$

O cheie de criptare *K* este folosită pentru a crea 16 subchei $k_1, k_2, ..., k_{16}$ având câte 48 de biți.

Funcția *F*, care intră în componența funcției f_k, are ca parametri un bloc de 32 de biți și o subcheie de criptare. După cum se observă, unui bloc de date, care a trecut de permutarea inițială, îi este aplicată funcția f_k de 16 ori.

Pașii care trebuie aplicați de 16 ori consecutiv, pentru a cripta un bloc de 64 de biți, sunt următorii:

- unui bloc din mesajul inițial îi este aplicată permutarea inițială și apoi este împărțit în două jumătăți L_0 și R_0;
- se generează cele 16 subchei din cheia de criptare *K*, prin aplicarea unei permutări inițiale urmată de 16 deplasări circulare la stânga cu 1 bit ale blocului rezultat, fiecare deplasare fiind urmată de o permutare a ultimilor 48 de biți și rezultând, la fiecare pas *i*, o subcheie de criptare k_i;
- algoritmul pentru funcția de criptare f_k este următorul:
  - ♦ la fel ca în cazul algoritmului *S-DES,* la fiecare pas *i* avem:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$$

  - ♦ funcția *F* are doi parametri: blocul R_{i-1} care are 32 de biți și subcheia k_i. Cei 32 de biți sunt expandați la 48 după care sunt permutați și se aplică operatorul *SAU-exclusiv* între aceștia și subcheia k_i. Rezultatul este împărțit în 6 blocuri de câte 8 biți, fiecare fiind prelucrat folosindu-se 8 *S-matrice* distincte. Pentru cele 8 *S-matrice* intrarea are 6 biți, iar la ieșire sunt furnizați 4. Fiecare *S-matrice* are 4 linii și 16 coloane și fiecare linie conține o permutare distinctă a șirului format din numerele de la 0 la 15. Primul și ultimul bit din fiecare grup de 6 biți sunt folosiți pentru a

identifica numărul liniei din *S-matricea* corespunzătoare, iar ceilalți 4 sunt folosiți pentru numărul coloanei. Toate grupurile de 4 biți obținute din *S-matrice* în același mod ca și în cazul metodei *S-DES* formează un bloc de 32 de biți, ai cărui biți sunt în final permutați și reprezintă rezultatul funcției *F*;

- ♦ între rezultatul funcției *F* și jumătatea stângă L_{i-1} se aplică operatorul *SAU-exclusiv*;

- cu ajutorul funcției *SW*, cele două jumătăți ale blocului rezultat se interschimbă:

$$(L_i, R_i) \rightarrow (R_i, L_i)$$

- la sfârșitul celor 16 iterații, cele două jumătăți rezultate formează un număr asupra căruia se aplică inversa permutării inițiale, obținându-se blocul criptat.

Detalii

Algoritmul de criptare *DES* este un caz particular al clasei de tehnici de criptare numite criptări *Feistel*, care folosesc mai multe blocuri de date. Aceste tehnici folosesc alternativ mai multe funcții de *difuzie* și *confuzie*.

Funcțiile de difuzie au ca scop construirea de relații statistice între mesajul inițial și cel criptat, cât de complex este posibil cu scopul de a proteja mesajul împotriva atacurilor statistice care au ca scop construirea cheii de criptare. În procesul de criptare folosind blocuri de date, acest lucru se realizează prin permutarea elementelor blocului înainte de aplicarea unei funcții.

Funcțiile de confuzie au ca scop construirea de relații statistice între mesajul criptat și cheia de criptare, care să fie cât de complexe posibil. Aceasta se realizează prin utilizarea unui algoritm complex de substituție.

Un algoritm *Feistel* folosește blocuri de intrare având lungimea un număr par care este împărțit în două jumătăți L_i și R_i, și o subcheie k_1, derivată dintr-o cheie de criptare *K*. Cele două jumătăți sunt procesate de *n* ori, după care sunt recombinate pentru a forma blocul de ieșire (criptat). Fiecare operație de procesare are aceeași structură.

Un bloc *L* este substituit cu rezultatul unei expresii de forma $L \oplus F(R, k)$. După substituție, cele două jumătăți se interschimbă.

Algoritmii de criptare *Feistel* depind de:

- *mărimea blocului de intrare*: cu cât blocul de intrare este mai mare, cu atât securitatea datelor crește, dar crește și timpul alocat de calculator pentru procesare; de obicei se folosesc blocuri de 64 de biți;
- *mărimea cheii de criptare*: cu cât cheia de criptare este mai mare, cu atât securitatea datelor crește, dar crește și timpul necesar pentru prelucrare; de obicei se folosesc blocuri de 64 de biți;
- *numărul de iterații*: de obicei se folosesc 16 iterații în procesul de criptare; acest număr reprezintă un compromis rezonabil între securitate și viteză de procesare;
- *generarea subcheilor*: cu cât complexitatea subcheilor generate este mai mare, cu atât crește securitatea datelor;
- *funcția F*: această funcție este foarte importantă, deoarece, în general, aceasta nu este o funcție liniară și cu cât este mai complexă, cu atât securitatea datelor este mai mare.

După cum se observă, creșterea complexității oricăreia dintre cele 5 componente enumerate mai sus duce la creșterea timpului de procesare necesar pentru un calculator.

O facilitate interesantă și foarte utilă a algoritmilor de criptare *Feistel* este aceea că algoritmul de decriptare este același cu cel de criptare, singura diferență fiind aceea că subcheile obținute din cheia de criptare sunt folosite în ordine inversă.

Pentru algoritmul *DES* se folosesc diferite modele pentru a asigura diferite tipuri de securitate. Cel mai simplu model poartă numele de ***ECB*** (***E**lectronic **C**ook**B**ook*). Acest model criptează fiecare bloc de 64 de biți de date independent, folosind aceeași cheie. În consecință, aceleași două blocuri de date identice sunt criptate la fel, folosind acest model. Modelul *ECB* este folosit, de obicei, pentru a cripta un singur mesaj scurt care poate conține, de exemplu, o cheie de criptare.

Cel de-al doilea model este ***CBC*** (***C**ipher **B**lock **C**haining*). Fie P_i și C_i ale *i*-lea blocuri de date necriptate și criptate și E_k funcția de criptare cu cheia *k*, atunci:

$$C_1 = E_k(P_1 \oplus IV),\ C_n = E_k(C_{n-1} \oplus P_n),\ n \geq 2,$$

unde *IV* este un vector de inițializare care trebuie protejat în același mod în care este protejată și cheia de criptare. Cu ajutorul acestui model, două blocuri de date de intrare identice sunt criptate diferit. Acest model este folosit pentru a cripta mesaje lungi de date.

Dacă notăm cu D_k funcția de decriptare care folosește cheia *k*, atunci formula de decriptare a unui mesaj este:

$$\begin{aligned} C_{n-1} \oplus D_k(C_n) &= C_{n-1} \oplus D_k(E_k(C_{n-1} \oplus P_n)) \\ &= C_{n-1} \oplus C_{n-1} \oplus P_n = P_n. \end{aligned}$$

Cu toate că *DES* este un algoritm care criptează blocuri de date, acesta poate fi trasformat într-un algoritm care criptează fluxuri de date, folosind ***CFM*** (***C**ipher **F**eedback **M**ode*).

Să presupunem că dorim să transmitem un flux de caractere criptat care constă în *j* biți pe caracter (de obicei se folosesc 8 biți).

Dacă avem un vector *IV* de inițializare cu lungimea de 64 de biți, $S_j(B)$ sunt cei mai din stânga *j* biți ai blocului *B* și P_i, C_i au semnificația de mai sus, atunci fiecare caracter criptat este obținut astfel:

$$C_i = P_i \oplus S_j(E_k(B_i)),\ B_i = (B_{i-1}[(j+1):64], C_{i-1}),\ B_1 = IV,$$

unde $(B[(j+1):64], C)$ reprezintă un bloc de 64 de biți format prin deplasarea circulară la stânga cu *j* biți a blocului *B*, după care ultimii *j* biți din blocul B sunt înlocuiți cu blocul *C* (numit *registru de deplasare cu j biți*).

Operația de decriptare a unui astfel de mesaj este similară procesului de criptare, singura deosebire fiind aceea că se inversează rolurile lui C_i și P_i, deci P_i se transformă în C_i și invers.

Claudiu Soroiu este redactor al GInfo. Poate fi contactat prin e-mail la adresa csoroiu@yahoo.com.