Tvorba šifrátoru DES

1. 4. 2017

1. Obsah

Obsah

2. Úvod

Zadání

Úkolem tohoto zadání je vytvořit program v programovacím jazyce Python, který umožňuje šifrovat a dešifrovat vstupní data uživatele pomocí algoritmu DES. Zakázáno nám bylo používat již existující knihovny, které podporují algoritmus DES. V další části se zaměříme i na 3DES šifrátor.

Cílem projektu je vytvořit funkční DES šifrátor, který zabezpečí uživatelova data v podobě jednoduchého textu nebo textového souboru (s příponou „.txt“) pomocí zadaného klíče tak, aby případný útočník nemohl data jednoduše přečíst a znát tedy jejich obsah.

Pomůcky

K realizaci šifrátoru se použije již zmiňovaný programovací jazyk Python. Tento programovací jazyk je velice jednoduchý a lze psát v téměř jakémkoliv textovém editoru. Jeho výstupem musí být, akorát, soubor s příponou „.py“.

Většina našeho týmu bude využívat k programování projektu rozšiřující balíček Python pro Visual Studio 2012, nebo známý textový editor PSPad.

Pro překlad zdrojového kódu musí být na příslušné stanici být nainstalovaný kompilátor pro Python, který je volně dostupný na https://www.python.org/downloads/ .

Tento soubor obsahuje pouhou dokumentaci realizace šifrátoru, bez praktických ukázek.

3. Teorie DES

Historie

V roce 1973 NBS (National Bureau of Standards, v současnosti NIST – National Institute of Standards and Technology) zveřejnil požadavek ve Federálním registru pro šifrovací algoritmus, který měl splňovat určité požadavky:

- Mít vysokou úroveň bezpečnosti na úkor použití relativně malého klíče pro šifrování a dešifrování
- Snadno pochopitelný
- Nezávislý na utajení algoritmu
- Adaptovatelný a ekonomický
- Efektivní a přenositelný

NBS čekalo na odpověď, která dorazila až o rok později od IBM, který doporučil algoritmus zvaný „Lucifer“, vyvíjený Horstem Feistelem a jeho kolegy. Tento algoritmus byl upraven NSA (National Security Agency), a tak vzniknul DES. Standardizován společností ANSI (American National Standard) pod jménem ANSI X3.92, nebo také DEA (Data Encryption Algorithm)

Vlastní funkcionalita DES

Vlastní části algoritmu:

- Rozdělení vstupu na 64 bitové bloky (8 oktetů)
- Iniciální permutace bloků
- Rozdělení bloků na 2 32 bitové části: **L** a **R** (levá a pravá)
- Permutace a substituce - 16x se opakuje (nazýváno **Rundy**)
- Spojení levé a pravé části a následná inverzní permutace

V rozdílné části algoritmu se provádí tzv. Key Scheduler, nebo-li vytvoření klíčů.

Vytvoření klíčů

DES má na vstupu zprávu, kterou chceme zabezpečit a klíč, který má velikost 64 bitů. Tento klíč se použije k šifrování, i k dešifrování.
DES pracuje s bloky o velikosti 64 bitů a používá klíč velikosti 56 bitů. Klíče jsou ukládány s velikostí 64 bitů, každý osmý bit se však nepoužije (tzn. bity číslo 8, 16, 24, 32, 40, 48, 56, 64).

Z 64 bitového klíče se vytvoří 16 podklíčů pomocí permutace, každý podklíč má velikost 48 bitů. K mermutaci použijeme tabulku PC-1. V této tabulce je první číslo 57, to znamená, že 57. bit z našeho klíče se použije jako 1. bit podklíče, 49. bit se použije jako druhý bit podklíče, takto pokračujeme až k poslednímu bitu podklíče, tím bude 4. bit klíče. V podklíči se objeví pouze 56 bitů z klíče.

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

Příklad:
64 bitový klíč K:
K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

Permutovaný klíč K+ podle tabulky s 56 bity:
K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Tento klíč K+ rozdělíme na 2 poloviny o velikosti 28 bitů. Levou polovinu označíme C_0 a pravou D_0.

Získáme:
C_0 = 1111000 0110011 0010101 0101111
D_0 = 0101010 1011001 1001111 0001111

Z C_0 a D_0 nyní vytvoříme 16 bloků, které jsou na sobě závislé. Každý blok C_n a D_n je vytvořený z bloku C_{n-1} a D_{n-1} bitovým posunem vlevo (Každý bit se posune vlevo, první byt je převeden na konec bloku) podle této tabulky:

Iterace	Bitový posun o
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

Příklad:
C_1 = 1110 0001 1001 1001 0101 0101 1111
D_1 = 1010 1010 1100 1100 1111 0001 1110
C_2 = 1100 0011 0011 0010 1010 1011 1111
D_2 = 0101 0101 1001 1001 1110 0011 1101

Takto pokračujeme až do 16. Iterace.

Poslední krok je vytvoření 48 bitového klíče pomocí tabulky PC-2 z předchozího 56 bitového klíče.

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

Stejně jako u první permutace číslo 14 znamená, že 14 bit z klíče C1D1 je použitý jako první byt našeho 48 bitového klíče a takto postupujeme dále až k 32. bitu klíče jako poslednímu v podklíči.

Příklad:
C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

Po užití permutace z tabulky PC-2 dostaneme K1 ve tvaru
K_1 = 000110 110000 001011 101111 111111 000111 000001 110010¨

C_2D_2 = 1100001 1001100 1010101 0111111 0101010 1100110 0111100 0111101
K_2 = 011110 011010 111011 011001 110110 111100 100111 100101

Takto pokračujeme dál, až budeme mít všech 16 klíčů.

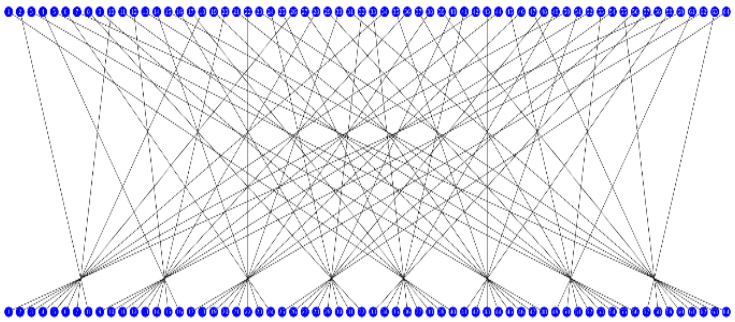
Initial permutation (IP)

Pomocí vstupní permutace se 64 bitové bloky rozdělí na 8 bitové bloky. Toto rozdělení nemá žádný vliv na zvýšení bezpečnosti samotné šifry, ale zvyšuje efektivnost provádění.

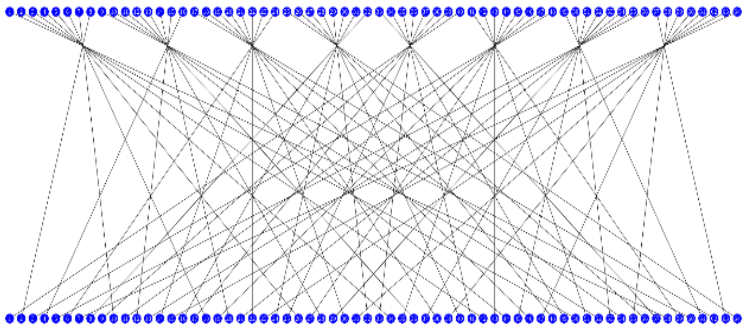
V době formování šifry DES se řešila otázka efektivnosti a pro zrychlení provádění se zavedla permutace, která vstupní blok rozdělila na 8 menších. Díky tomu bylo možné naráz posílat celý registr, protože sběrnice byly 8 bitové. To znamená, že po osmi provedeních bylo možné přejít k samotné šifře. Toto řešení by se dalo vynechat, ale následkem by byly velice složité a drahé hardwarové okruhy.

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

Před započetím rundování se provede vstupní permutace a po poslední rundě se provede inverzní permutace ke vstupní. Permutace je dána podle statické tabulky tzn. první bit výstupu bude 58 bit vstupu, druhý bit výstupu bude 50 bit vstupu atd.



Vstupní permutace



Výstupní permutace

Rozdělení do 32 bitových bloků

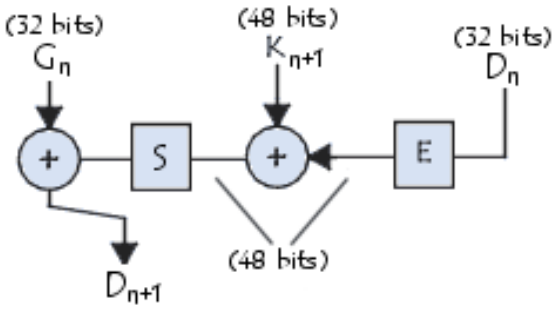
Po dokončení počáteční permutace se 64 bitový blok rozdělí do dvou 32 bitových bloků, levý blok L a pravý blok R. V levém bloku tedy budou bity, které mají sudou pozici a v pravém bloku budou bity s lichou pozicí. Na počátku jsou tyto strany nazvané L_0 a R_0.

L_0							
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

R_0							
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

Rundy

Po rozdělení 64 bitové, permutované, zprávy na 2 32 bitové bloky (levý L_i, respektive pravý R_i) se provede 16 „rund“, které mají následující funkce:



Expanzní funkce

32 bitový blok R_i (i ∈ <0;15>) se dále rozšiřuje na 48 bitů díky expanzní tabulce (označováno E), ve které je všech 48 bitů prohozeno, a 16 z nich jsou duplikace.

E	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

Operace XOR s klíčem

Po expanzi z 32 bitového (R_i; i ∈ <0;15>) bloku na 48 bitový (R_{iE}) se provede exkluzivní OR operace, kde na vstupu je R_{iE} a vytvořené klíče (K_i; i ∈ <0;16>) v jedné z přechozích funkcí. Výsledkem je zase 48 bitový blok R_i (pozor! Ne počáteční R_i, které je přivedeno na vstup XOR funkce)

S-Box

R_i je dále rozděleno na 8 6 bitových bloků, označovány R_{0i}. Každý z bloků R_{0i} se prožene funkcí výběru (označováno jako Substituční box, nebo také kompresní funkce), obecně označováno jako S_i (i ∈ <0;7>). O každý z těchto bloků R_{0i} se stará jiná substituční tabulka S_i.

Substituční tabulka se skládá ze 4 řádků a 16 sloupců, kde každým vstupem je 4 bitové číslo. 6 bitové vstupní číslo R_{0i} specifikuje pod kterým řádkem, respektive sloupcem, se má vybírat výstup. Kombinace prvního a posledního bitu z R0i udávají číslo řádku (číslo 0 – 3) a kombinace ostatních bitů (2., 3., 4. a 5. bit, respektive číslo 0 – 15) označují číslo sloupce, ze kterého se má vybírat. Výstupem je tady 4 bitové číslo.

Substituční tabulky

S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	1	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	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	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

P-Box

Ze získaných 32 bitů (8 × 4 bit blokový výstup z každého S-Box) se dle uvedené permutační tabulky permutuje vstup. Výstupem je R_{0i}, které je velké 32 bitů.

P							
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

Reverzní iniciální permutace

Na konci všech rund, tyto dva bloky L_{16} a R_{16} jsou znovu spojeny a podrobeny inverzní iniciální permutaci dle následující tabulky.

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

Výstupem je cílený 64 bitový šifrový text.

4. Zdroje a autoři projektu

Autoři

David Kolaja – Team Leader, S-Box, P-Box
Michael Jelínek – Iniciální Permutace, Expanzní funkce
David Hirš – Vytvoření klíčů
Michal Loskot – Rozdělení 64 bitového vstupu na 32 bitové bloky, XOR s klíčem

Použité zdroje

- http://www.nku.edu/~christensen/DESschneier.pdf
- http://ccm.net/contents/134-introduction-to-encryption-with-des
- http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm