

SEMESTRÁLNÍ PRÁCE Z PŘEDMĚTU KIV/BIT

IMPLEMENTACE ŠIFRY SAFER K-64

Jméno: Jakub Záruba

Osobní číslo: A13B0476P

E-mail: eflyax@students.zcu.cz

Datum: 2. května 2016

Obsah

1	O šifře SAFER	3
2	Popis algoritmů	4
2.1	Šifrování	4
2.2	Produkce podklíčů	6
2.3	Dešifrování	6
3	Dokumentace	8
3.1	Programátorská dokumentace	8
3.2	Uživatelská dokumentace	9
4	Použitá literatura a zdroje	10
5	Závěr	11

1 O šifře SAFER

Šifra SAFER K-64 (Secure And FAST Encryption Routine, s 64bitovým klíčem), je bloková šifra s 64bitovým plaintextem. Šifra SAFER k-64 spadá do rodiny šifer SAFER, které navrhnul James Massey (jeden z návrhářů šifry IDEA). Poprvé byla publikována v roce 1993. Všechny algoritmy z rodiny SAFER jsou nepatentované a jsou k dispozici pro neomezené použití.

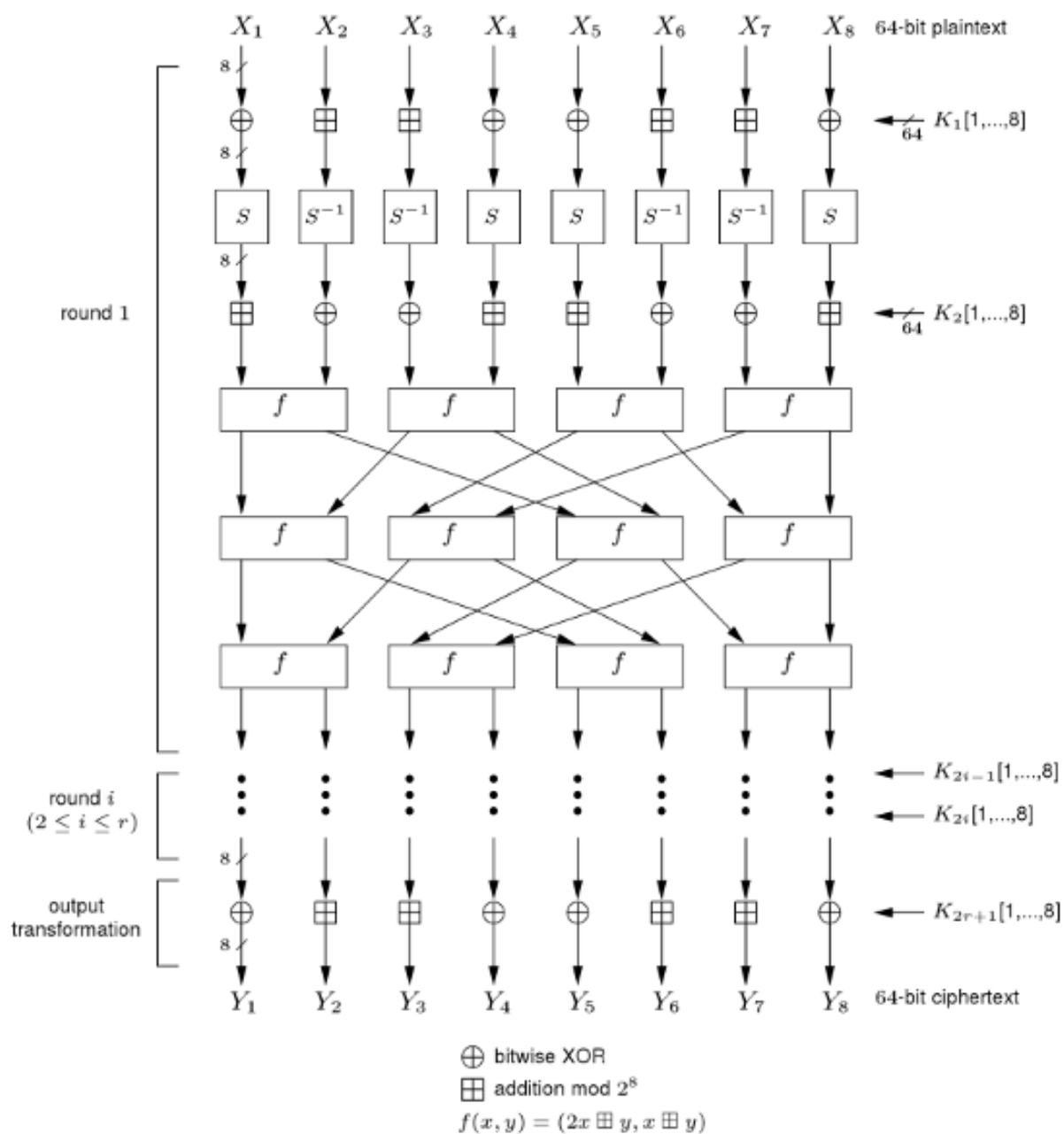
2 Popis algoritmů

Šifra používá r identických šifrovacích kol (rounds). Tato šifra standardně používá 6 rounds, počet může být volitelný. Maximum je však 10. Pro zadaný 64 bitový klíč se vytvoří $2r + 1$ podklíčů, které mají také 64 bitů. 2 klíče se vždy použijí v jednom šifrovacím kole ($2r$) a poslední klíč ($+1$) se použije pro výstupní transformaci. Uživatelem zvolený vstup a klíč je dlouhý 8 bajtů (64 bitů).

2.1 Šifrování

- **Vstup:** $r, 6 \leq r \leq 10$; 64 bitový text $M = m_1 \dots m_{64}$; klíč $K = k_1 \dots k_{64}$
- **Výstup:** 64 bitový šifrovaný text (blok) $Y = (Y_1, \dots, Y_8)$

1. Výpočet podklíčů K_1, \dots, K_8 , viz kapitola 2.2
2. $(X_1, X_2, \dots, X_8) \leftarrow (m_1 \dots m_8, m_9 \dots m_{16}, \dots, m_{57} \dots m_{64})$.
3. Pro i od 1 proved' akce: (XOR-addition, S-box, XOR-addition, 3x lineární vrstvy)
4. (a) Pro $j = 1, 4, 5, 8$: $X_j \leftarrow X_j \oplus K_{2i-1|j|}$
Pro $j = 2, 3, 6, 7$: $X_j \leftarrow X_j \boxplus K_{2i-1|j|}$
(b) Pro $j = 1, 4, 5, 8$: $X_j \leftarrow S[X_j]$, Pro $j = 2, 3, 6, 7$: $X_j \leftarrow S_{inv}[X_j]$.
(c) Pro $j = 1, 4, 5, 8$: $X_j \leftarrow S[X_j] \boxplus K_{2i|j|}$. Pro $j = 2, 3, 6, 7$: $X_j \leftarrow X_j \oplus K_{2i|j|}$.
(d) Pro $j = 1, 3, 5, 7$: $(X_j, X_{j+1}) \leftarrow f(X_j, X_{j+1})$
(e) $(Y_1, Y_2) \leftarrow f(X_1, X_3)$, $(Y_3, Y_4) \leftarrow f(X_5, X_7)$
 $(Y_5, Y_6) \leftarrow f(X_2, X_4)$, $(Y_7, Y_8) \leftarrow f(X_6, X_8)$
Pro j od 1 do 8 proved' : $X_j \leftarrow Y_j$
(f) $(Y_1, Y_2) \leftarrow f(X_1, X_3)$, $(Y_3, Y_4) \leftarrow f(X_5, X_7)$
 $(Y_5, Y_6) \leftarrow f(X_2, X_4)$, $(Y_7, Y_8) \leftarrow f(X_6, X_8)$
Pro j od 1 do 8 proved' : $X_i \leftarrow Y_j$
5. Výstupní transformace:
Pro $j = 1, 4, 5, 8$: $Y_j \leftarrow X_j \oplus K_{2r+1|j|}$.
Pro $j = 2, 3, 6, 7$: $Y_j \leftarrow X_j \boxplus K_{2r+1|j|}$



Obrázek 1: Schéma šifrování

2.2 Produkce podklíčů

- **Vstup:** 64 bitový klíč $K = k_1 \dots k_{64}$; počet šifrovacích kol r
- **Výstup:** 64 bitové podklíče K_1, \dots, K_{2r+1}

1. Nechť $R[i]$ představuje 8 bitové úložiště dat a nechť $B_i[j]$ představuje bajt j v B_i
2. $(R[1], R[2], \dots, R[8]) \leftarrow (k_1 \dots k_8, k_9 \dots k_{16}, \dots, k_{57} \dots k_{64})$
3. Pro i od 2 do $2r+1$ proved':
 - (a) Pro j od 1 do 8 proved': $R[j] \leftarrow (R[j] \leftarrow 3)$
 - (b) Pro j od 1 do 8 proved': $K_i[j] \leftarrow R[j] \boxplus B_i[j]$

Pro vstupní klíč *12345678* s 6 iteracemi bude tabulka podklíčů vypadat takto:

Tabulka 1: Vizualizace přiřazení podklíčů k iteracím

hmm	Použité podklíče							
1. iterace	1	2	3	4	5	6	7	8
	86	171	107	70	174	136	205	142
2. iterace	73	63	165	151	242	55	8	134
	193	200	175	193	24	16	201	57
3. iterace	73	202	136	252	164	213	12	187
	202	234	152	218	15	121	155	118
4. iterace	134	248	29	81	227	150	203	220
	107	201	246	233	195	31	43	244
5. iterace	163	111	166	106	97	90	148	32
	177	148	235	74	225	214	139	196
6. iterace	101	85	224	107	98	120	180	114
	13	9	35	74	238	87	33	67
Transformace	15	153	61	84	192	14	7	65

2.3 Dešifrování

Proces dešifrování používá pro zadaný klíč K stejné hledání podklíčů, jako při šifrování. Proces začíná s se vstupní transformací s klíčem K_{2r+1} k výstupní transformaci. Veškeré modulární

součty jsou nahrazeny modulárním odečítáním. Funkce f v lineárních vrstvách jsou nahrazeny jejich inverzními funkcemi: $f_{inv}(L,R) = (L - R, 2R - L)$ s odečítáním mod 256 v tříkrokových sekvencích.

3 Dokumentace

3.1 Programátorská dokumentace

- **void SAFER_K_64_encryption(. . .)**

Provádí šifrování vstupního textu se zadaným klíčem.

- **void SAFER_K_64_key_schedule(. . .)**

Provádí rozklad klíče na podklíče, které se používají pro šifrování/dešifrování.

- **void generate_S_boxes(. . .)** Generuje tabulky konstant, které se používají při rozkladu klíče na podklíče. Funkce přijme v parametru tabulky (pole) S a S_{inv} s velikostí 512. Dle dokumentace šifry se aplikuje následující algoritmus, pro naplnění tabulek:

```
short g = 45, i, t;
S[0] = 1, S_inv[1] = 0;
for (i = 1; i <= 255; i++) {
    t = (short) ((g * S[i - 1]) % 257);
    S[i] = t;
    S_inv[t] = (uchar) i;
}
S[128] = 0, S_inv[0] = 128;
```

Obě tabulky ($S - Boxy$) jsou vůči sobě inverzní. Obecně platí, že v tabulce S na indexu \mathbf{X} s hodnotou \mathbf{Y} , je v tabulce S_{inv} na indexu \mathbf{Y} hodnota \mathbf{X} .

- **void f(uchar x, uchar y, uchar *X, uchar *Y)**

Jedná se o implementaci lineární transformace, použitou ve 3 vrstvě šifrovacího algoritmu. Tato funkce je také známá jako pseudo-Hadamardova transformace.

- **void f_inv(uchar L, uchar R, uchar *l, uchar *r)**

Jedná se o inverzní funkci k funkci f . Tato inverzní transformace je využívána při dešifrování.

- **void SAFER_K_64_decryption(. . .)**

Provádí dešifrování dle zadaného vstupu a klíče.

- **int check_user_inputs(. . .)**

Validuje vstupy zadané od uživatele (vstupní text a šifrovací/dešifrovací klíč).

- **int main(int argc, char *argv[])**

Hlavní procedura programu. Volá podprogram pro validaci vstupů, dle zadaných vstupů provede šifrování/dešifrování. Veškeré výstupy programu se vypisují do konzole.

3.2 Uživatelská dokumentace

Program jsem vytvořil v jazyce C, je spustitelný pod operačními systémy Windows a Linux.

Program se spouští s 3 parametry:

```
./safer <akce> <text> <klic>
```

akce: hodnoty (e/d), e = šifrování, d = dešifrování

text: text, který bude program šifrovat/dešifrovat

klic: klíč, který se použije k šifrování/dešifrování

Například pro zašifrování slovo *kočka* s heslem *123456* použijete:

`./safer e kočka 123456`

4 Použitá literatura a zdroje

- Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. Handbook of Applied Cryptography. vyd: CRC Press, 1996. 755 s. ISBN 0-8493-8523-7.
- In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11. 12. 2006, last modified on 12.12 2015 [cit. 20.4 2016].
Dostupné z: <https://en.wikipedia.org/wiki/SAFER>

5 Závěr

Podařilo se mi úspěšně implementovat šifru SAFER K64, schopnou šifrovat a dešifrovat zadané vstupy. Při implementaci a studování algoritmů jsem využil vědomosti z přednášek, které mi při vývoji velmi pomohly. Práci jsem vytvářel na operačním systému Ubuntu 14.04. Nastudování potřebných materiálů, implementace a napsání dokumentace mi zabralo zhruba 55 hodin.