Návrh a kryptoanalýza šifier Zadanie 4

Pavol Sobota
16.10.2022

1 Úvod

Implementácia SPN šifrátora bola napísaná v jazyku C. Na kompilovanie sme použili kompilátor CLANG s verziou 13.0.0 (clang-1300.0.29.3). Počítač, ktorí spúšťal implementáciu mal parametre:

- Procesor: Apple M1.
- RAM: 8 GB.
- GPU: Apple M1 7-jadrová GPU.

2 Riešenie

V nasledujúcich sekciách priblížime funkcionalitu jednotlivých komponentov implementácie.

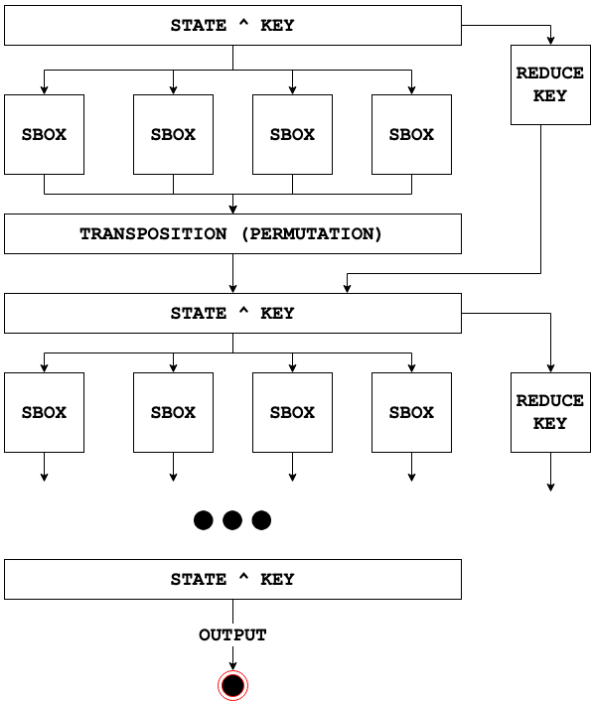

Figure 1: Architektúra blokového šifrátora SPN

2.1 Architektúra

Vstupom do šifrátora bola 16 bitová hodnota **u_int16_t**. Kľúč bol rovnakej dĺžky. Algoritmus šifrovania zbehol 100 krát pre vytvorenie 100 súborov. V každom súbore bol 8 krát zašifrovaný plaintext **0x0000** až **0xffff** pre 8 rôznych kľúčov. SPN šifrátor mal R kôl, kde R mohlo nadobudnúť hodnotu od 4 až teoreticky po 20,922,789,888,000 (počet permutácií 16 prvkovej množiny). Prvý kľúč pred začiatkom šifrovania bol nastavený na **0xabcd**. Po každej iterácii SPN sa zvýšil o 1. Na začiatku SPN šifrátora bol vstup sčítaní s kľúčom bez transformácie pomocou operácie XOR.

2.2 Funkcia SBOX

Pred vstupom do SBOX-ov bola šifrovaná hodnota rozdelená na 4 bitové vstupy. V SBOX-e boli substituované za novú hodnotu. Substitučná tabuľka bola vygenerovaná náhodne a bola rovnaká pre všetky SBOX-y. Po výstupe z SBOX-ov boli hodnoty znovu spojené pomocou XOR.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	3	6	2	F	5	8	9	D	E	C	B	A	1	0	7

Table 1: Substitučná tabuľka SBOX-ov

2.3 Permutačná funkcia

Následne bola celá 16 bitová hodnota permutovaná podľa nasledovnej tabuľky. Tabuľka bola vygenerovaná pomocou príkazu zo zadania a AIS ID 92202.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F	2	B	4	9	0	7	8	6	A	1	B	E	3	C	5

Table 2: Permutačná tabuľka

2.4 Transformačná funkcia pre podkľúč

Pri vytváraní podkľúčov sme sa rozhodovali ako budeme postupovať aby bolo možné transformáciu parametrizovať. Najpr sme experimentovali s lineárnym posunom bitov na kľúči, ktorý bol väčší ako veľkosť bitov vstupu. Avšak pri väčšom počte kôl SPN by sme museli ten istý posun opakovať viac krát. Teda niektoré kľúče by boli použité n-krát a niektoré (n-1)-krát. To znamená že podmienka difúzie by nebola dostatočne splnená, šifrovaný text by nebol rovnako závislý na všetkých bitoch kľúča. Na výrobu podkľúčov sme teda využili taktiež permutácie a to pomocou výberu z lexikograficky zoradenej tabuľky permutácií. Tak že v k-tom kole behu SPN šifrátora bola ako kľúč použitá k-ta permutácia z tabuľky. Napríklad pre kľúč `1010101010101010` by bola permutácia v prvom kole `1010101010101001`.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	A	B	C	D	F	E
0	1	2	3	4	5	6	7	8	9	A	B	C	E	D	F
...															
F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Table 3: Lexikograficky zoradené permutácie

3 Testovanie a Analýza

Na otestovanie neodlíšiteľnosti od náhodných postupností sme zašifrované súbory podrobili NIST testom z programu ParanoYa Academic 2010. Pre porovanie sme zvolili 3 rôzne batch-e testovacích súborov. Odlišovali sa počtom kôl SPN, ktoré ich generovali. Zvolili sme 4, 5 a 10 počet kôl. Neúspešnosť testov spolu s časom jednotlivých behov šifrovania sú uvedené v nasledujúcej tabuľke.

Test	Neúspešnosť na hladine 0,01		
	NR=4	NR=5	NR=10
Frequency (monobit)	80%	79%	3%
Frequency within a block	100%	100%	1%
Runs test	16%	24%	0%
Test for lognest run of one in block	92%	94%	0%
Binary matrix rank	100%	1%	0%
Discrete Fourier transform (spectral)	100%	100%	0%
Non-overlapping t emplate matching	10,5%	2,1%	1%
Overlapping template matching	11%	3%	1%
Maurer's "Universal statistical test"	99%	13%	3%
Linear complexity	1%	1%	3%
Approximate entropy test	lr	5%	0%
TIME	18s	18s	18s

Table 4: Výsledky NIST testov pre jednotlivé SPN šifrátory

4 Záver a Komentár

Úspešne sme navrhli blokovú šifru s dostatočnými konfúznymi a difúznymi vlastnosťami. Po otestovaní NIST testami sme zhodnotili že na dosiahnutie podmienok je potrebné vykonať 10 kôl SPN šifrátora. Čas, ktorý tento počet kôl trval v porovnaní s ostatnými SPN šifrátormi je zanedbateľný. Transformácia kľúča na podkľúče je vhodná aj keď nie ideálna. Keďže medzi jednotlivými kolami sa zmení iba pozícia 2 bitov. Lepšie by bolo využiť iné pozície permutácií v lexikografickom zozname. Napr pre zoznam dĺžky **n** a počet kôl dĺžky **NR**.

```
k=n/NR
key[0]=n[0*k]
key[1]=n[1*k]
.
.
.
key[NR-1]=n[NR-1*k]
```

5 Prílohy

Implementácia v jazyku C.
`https://github.com/Nnabuchodonozor/NKS-2022/tree/main/dj%20bloko`