Návrh a kryptoanalýza šifier - Zadanie 4

Peter Čuřík

17. októbra 2021

1 Substitučno-permutačná sieť

Na účely zadania bola skonštruovaná tzv. SP sieť podľa zdroja *tu*. Použitý S-box bol vygenerovaný podľa python príkazu

`numpy.random.RandomState(seed=int(ais_id)).permutation(16),`

kde `ais_id` je v mojom prípade číslo 91764. Výsledkom je permutácia (S-box), vyzerajúca nasledovne:

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

K danému S-boxu bol vypočítaný inverzný S-box (potrebný pre dešifrovanie):

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

Vstupy do tabuľky S-boxu sú postupnosti štyroch bitov a výstupmi sú substitúcie podľa tabuľky S-boxu.

2 Implementácia

Vlastné riešenie v jazyku C. Vytvorenie dvoch zdrojových súborov. Jeden šifruje pomocou jedného kľúča všetky možné plaintexty. Komplexita: 2^{16}. Druhý dešifruje pomocou všetkých kľúčov jeden ciphertext. Komplexita: 2^{32}.

Riešenie je schopné šifrovať otvorený text a následne výstup naspäť dešifrovať na pôvodný otvorený text. Príklad je ukázaný nižšie.

ENCRYPT:		DECRYPT:	
plaintext:	`0xFFFF`	ciphertext:	`0x81C7`
kľúč:	`0xC825881F`	kľúč:	`0xC825881F`
ciphertext:	`0x81C7`	plaintext:	`0xFFFF`

Kód je rozdelený do piatich funkcií: decrypt (resp. encrypt), key addition, permutation, sbox, main funkcia.

Šifrovanie, resp. dešifrovanie prebieha v štyroch kolách. Pre každé kolo je generovaný podkľúč. A to pomocou extrakcie 16-tich bitov z hlavného, 32-bitového kľúča. Prvý podkľúč je použitie 1. - 4. štvorice bitov, druhý podkľúč 2. - 5. štvorica bitov a tak ďalej.

Key addition spočíva v operácii XOR podkľúča a aktuálneho stavu šifry.

Pri S-box operácii je používaná S-box tabuľka. Vstupuje sa do nej pomocou indexu, čo je hodnota jednotlivých štvoríc bitov v 16 bitovom stave šifry. Výstupom z tabuľky je substitúcia. Tieto štyri, štvorbitové substitúcie sú spojené pomocou XOR operácie do 16-tich bitov, reprezentujúcich nový stav šifry.

Pri permutácii si všímame určitý vzorec aplikovaný pri operácii: prvá štvorica bitov po permutácii vznikne ako výsledok spojenia prvých bitov všetkých štvoríc bitov. Druhá štvorica bitov po permutácii vznikne ako

výsledok spojenia druhých bitov všetkých štvoríc bitov. A tak ďalej. Bitovými operáciami bolo možné tento postup aplikovať do kódu. Grafická reprezentácia ukazuje myšlienku na jednoduchom príklade:

Vstup:
`1001 1001 0010 0000`

Výstup:
`1100 0000 0010 1100`

2.1 Opitmalizácie

V zadaní boli použité viaceré opitmalizačné techniky za účelom dosiahnutia maximálnej možnej rýchlosti kompilácie kódu.

2.1.1 Kompilácia

Kompilovanie s `gcc`. Používanie prepínača `-O3`, ktorý spočíva v aktivovaní všetkých optimalizácií prepínača `-O2` a zároveň v aktivovaní niektorých ďalších optimalizačných flagov.

2.1.2 Štruktúrovanie kódu

Zabaľovanie funkcionalít do samostatných C funkcií (spomínané už v úvode kapitoly). Nízkoúrovňová práca s bitmi iba pomocou bitových operácií (bitové posuny, AND masky, XOR operácie). Deklarácia premenných pomocou typov definovaných v knižnici `stdint.h` (šetrenie pamäte).

Bit slicing nebol použitý. Má zmysel až pri komplexnejších návrhoch s väčším počtom bitov. Rozbaľovanie cyklov nebolo použité. Prepínač kompilátora `gcc`, `-O3` je schopný optimalizovať kód lepšie, ak ostáva v cykloch.

3 Výsledky

3.1 SP sieť vlastného riešenia v C

Nižšie uvedené výsledky vznikli kompiláciou C kódu príkazom `gcc file.c -o spn -O3` a následným spustením programu príkazom `time ./spn`.

Šifrovanie všetkých plaintextov pomocou jedného kľúča:	0.098s (chyby merania: 0.070s - 0.304s)
Každé ďalšie spustenie spustiteľného súboru (cache):	0.005s (chyby merania: žiadne)
Dešifrovanie ciphertextu pomocou všetkých kľúčov:	0.186s (chyby merania: 0.091s - 0.259s)
Každé ďalšie spustenie spustiteľného súboru (cache):	0.005s (chyby merania: žiadne)

3.2 AES-128-ECB v softvéri OpenSSL

Po zadaní príkazu `openssl speed -evp aes-128-ecb` v softvéri OpenSSL bol získaný nasledovný výstup:

```
Doing aes-128-ecb for 3s on 16 size blocks: 106167653 aes-128-ecb's in 3.00s
```

Z výstupu vyplýva, že ide o 35 389 218 AES-128-CBC šifrovaní 16 bajtových blokov za sekundu. To je 283 113 744 šifrovaní 16 bitových blokov za sekundu.

Ak SP sieť použitá v tomto zadaní je schopná aplikovať 2^{16} šifrovaní 16 bitových blokov za 0.098s, potom platí, že za jednu sekundu by stihlo byť aplikovaných približne 668 729 šifrovaní. To je 423 násobne menší výkon, ako poskytuje OpenSSL pri šifre AES-128, v móde ECB.

Za predpokladu spúšťania SP siete z pamäte cache by sa rýchlosť podstatne zvýšila, z 668 729 šifrovaní za sekundu na 13 107 200 šifrovaní za sekundu. To je 21 násobne menší výkon, ako poskytuje OpenSSL pri šifre AES-128, v móde ECB.