



Semestrální práce

Bezpečnost v informačních technologiích

Ondřej Drtina
A17B0202P
drtinao@students.zcu.cz

Obsah

1	Zadání	2
1.1	Celé znění úlohy	2
2	Analýza problému	3
2.1	Zdroje informací	3
2.2	Posloupnost klíčových operací algoritmu	3
2.3	Operace key_expansion	4
2.4	Operace key_expansion_core	5
2.5	Operace sub_bytes	5
2.6	Operace shift_rows	5
2.7	Operace mix_columns	6
2.8	Operace add_round_key	6
3	Popis implementace	8
3.1	Použitý programovací jazyk, vývojové prostředí	8
3.2	Funkce key_expansion(unsigned char *original_key)	8
3.3	Funkce key_expansion_core(unsigned char *sequence, unsigned char it_index)	8
3.4	Funkce sub_bytes(unsigned char *data_block)	9
3.5	Funkce shift_rows(unsigned char *data_block)	9
3.6	Funkce mix_columns(unsigned char *data_block)	9
3.7	Funkce add_round_key(unsigned char *state, unsigned char *round_key)	9
3.8	Funkce encrypt_state(unsigned char *data_block, unsigned char *expanded_key)	10
4	Uživatelská příručka	11
4.1	Překlad programu	11
4.2	Spuštění v módu šifrování textu	11
4.3	Spuštění v módu šifrování souborů	11
5	Závěr	13
5.1	Funkčnost programu a jeho využitelnost v praxi	13
5.2	Problémy spojené s tvorbou programu	13
5.3	Zhodnocení, přínos úlohy pro řešitele	13

1 Zadání

V rámci semestrální práce z předmětu BIT jsem si zvolil standardní zadání. Mým cílem tedy bylo implementovat blokovou šifru AES. Na implementaci šifry jsou kladeny následující požadavky:

- výsledek musí být v hexadecimálním formátu
- musí být použit šifrovací mód ECB (Electronic Codebook)
- délka klíče a bloku musí být 128 bitů
- klíč bude volen uživatelem

Pro úplnost dodávám, že po domluvě s vyučujícím bylo možno si v rámci daného předmětu zvolit i jiná zadání semestrální práce.

1.1 Celé znění úlohy

Celé zadání úlohy je možno nalézt online zde.

2 Analýza problému

Po přečtení zadání úlohy bylo jasné, že její splnění bude vyžadovat znalost šifrovacího algoritmu AES¹. Zopakoval jsem si tedy základní kroky daného algoritmu, jejich popis je předmětem následujících podkapitol.

2.1 Zdroje informací

Vzhledem k notorické rozšířenosti daného šifrovacího algoritmu lze nalézt množství webových stránek, ze kterých lze získat informace potřebné k porozumění algoritmu.

Hlavním zdrojem informací pro mě byl web NIST Computer Security Resource Center poskytující kompletní specifikaci šifry AES v rámci dokumentu dostupného [zde](#). Dokument má 47 stránek a pochází z roku 2001. Pro implementaci šifry nebylo nutné číst celý dokument, zajímal jsem se zejména o kapitolu číslo 5, v rámci které jsou jednotlivé kroky algoritmu příkladně popsány. Z daného dokumentu také pocházejí obrázky 2.1 a 2.2.

Před čtením samotné specifikace algoritmu AES jsem základní informace o jednotlivých krocích algoritmu a jejich posloupnosti načerpal z videa umístěného na serveru YouTube - [zde](#). Video nahrál vlastník kanálu s názvem Computerphile a algoritmus v rámci videa popisuje pan doktor Mike Pound.

YouTube nabízí ve spojitosti s šifrou AES mnoho dalších videí, ze kterých je možno načerpat informace o jednotlivých krocích daného algoritmu.

2.2 Posloupnost klíčových operací algoritmu

Pro zašifrování textu či souboru pomocí algoritmu AES je třeba dodržet pevně definovanou posloupnost kroků. Každá z těchto operací má své označení, které pochází z anglického jazyka a během analýzy jsem se nesetkal s českou alternativou k názvům operací. Vzhledem k uvedené skutečnosti budou tedy i v této práci názvy operací uváděny v originálním znění.

První operace algoritmu je běžně označována jako `key_expansion`. V rámci tohoto kroku dojde k rozšíření klíče na 176B - prakticky tedy vznikne 10 nových klíčů, jelikož velikost původního klíče byla 16B (viz kapitola 2.3).

¹Advanced Encryption Standard - typ symetrické blokové šifry

Následně dojde k rozdělení informací, jež chceme zašifrovat, na bloky o velikosti 16 bytů. Na takové bloky následně aplikujeme pseudokód, který se nachází níže (viz 2.1).

```
1 add_round_key(blok_k_zasifrovani , expanded_key);
2
3 index_kola = 0;
4 for (; index_kola < 9; index_kola++){
5     sub_bytes(blok_k_zasifrovani);
6     shift_rows(blok_k_zasifrovani);
7     mix_columns(blok_k_zasifrovani);
8     add_round_key(blok_k_zasifrovani , expanded_key +
9         (16 * (index_kola + 1)));
10 }
11 sub_bytes(blok_k_zasifrovani);
12 shift_rows(blok_k_zasifrovani);
13 add_round_key(blok_k_zasifrovani , expanded_key + 160);
```

Kód 2.1: Posloupnost operací algoritmu AES

Jak je vidět, nejprve dojde k využití funkce `add_round_key`. Následuje cyklus, který 9x po sobě aplikuje funkce `sub_bytes`, `shift_rows`, `mix_columns` a `add_round_key`. Při volání funkce `add_round_key` lze pozorovat, že se postupně posouváme dále v poli `expanded_key`, jež obsahuje jednotlivé podklíče.

Algoritmus má celkem 10 iterací. Závěrem, tedy v rámci poslední iterace, proběhne aplikace operací `sub_bytes`, `shift_rows`, `mix_columns` a `add_round_key`. Zejména je důležité si povšimnout, že v rámci poslední iterace není, na rozdíl od předchozích iterací algoritmu, volána funkce `mix_columns`.

2.3 Operace `key_expansion`

Zajišťuje rozvinutí původního 16B klíče na 176B klíč. V podstatě tedy vytvoří 10 nových klíčů, které jsou postupně použity v jednotlivých iteracích algoritmu AES.

Při expanzi klíče vždy pracujeme s posledními čtyřmi vygenerovanými byty, na které je aplikována funkce XOR s prvními čtyřmi byty v daném 16B podklíči. Při dokončení generování podklíče (tedy každých 16B) rovněž musí dojít k volání funkce `key_expansion_core` (viz 2.4).

2.4 Operace `key_expansion_core`

Pracuje se 4 bytovou sekvencí, ve které nejprve otočí posloupnost jednotlivých bytů (tzv. levá rotace). Následuje nahrazení jednotlivých bytů pomocí substituční tabulky, stejně jako v případě operace `sub_bytes` (viz 2.5). Závěrem proběhne operace XOR mezi prvním bytem sekvence a prvkem ze substituční tabulky dostupné např. zde.

2.5 Operace `sub_bytes`

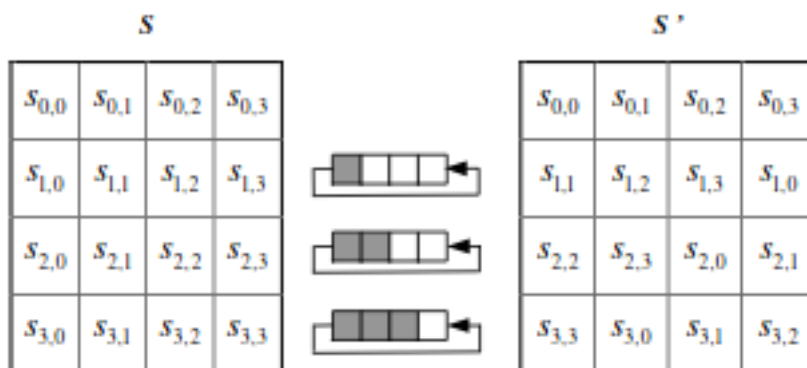
V rámci operace, jež je v dostupné literatuře označována jako `sub_bytes`, dojde prakticky pouze k nahrazení jednotlivých bytů 16-bytového bloku. Pomocí vyhledávací tabulky k algoritmu AES, jež je někdy označována také jako substituční tabulka, nalezneme hodnoty, kterými je potřeba jednotlivé byty nahradit. Danou tabulku lze pod názvem "Rijndael S-box" nalézt například zde.

2.6 Operace `shift_rows`

Daný krok zabezpečuje záměnu pořadí hodnot v jednotlivých řádcích matice, jež reprezentuje jeden datový blok (16 bytů). Při šifrování pomocí algoritmu AES je nutno v rámci tohoto kroku posunout hodnoty v řádcích matice vlevo o následující počet prvků:

- 0 - v případě první řádky = pořadí prvků první řádky zůstane nezměněno
- 1 - v případě druhé řádky = prvky ve druhé řádce budou posunuty o jeden prvek doleva
- 2 - v případě třetí řádky = prvky ve třetí řádce budou posunuty o dva prvky doleva
- 3 - v případě čtvrté řádky = prvky ve čtvrté řádce budou posunuty o tři prvky doleva

Prvky, které by se aplikací této operace dostaly mimo rozsah matice, se znovu objeví na druhé straně odpovídajícího řádku. Tento jev lze pozorovat na obrázku 2.1.



Obrázek 2.1: Posun učiněný krokem `shift_rows`

2.7 Operace `mix_columns`

V rámci této operace dojde k přepočítání veškerých prvků přítomných v matici. Pro každý prvek je provedeno několik operací XOR (= exkluzivní disjunkce) s ostatními hodnotami v daném sloupci. Vzorce, pomocí kterých lze zjistit výsledky kroku `mix_columns` jsou dostupné na obrázku 2.2. V případě prvků, u nichž je v rovnici 02 nebo 03, musí být použita odpovídající hodnota z vyhledávací tabulky (viz implementace - 3.6) nebo lze hodnoty vypočítat ručně.

Pozn.: tato operace není v posledním kroku algoritmu využita!

$$\begin{aligned}
 s'_{0,c} &= (\{02\} \cdot s_{0,c}) \oplus (\{03\} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\
 s'_{1,c} &= s_{0,c} \oplus (\{02\} \cdot s_{1,c}) \oplus (\{03\} \cdot s_{2,c}) \oplus s_{3,c} \\
 s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus (\{02\} \cdot s_{2,c}) \oplus (\{03\} \cdot s_{3,c}) \\
 s'_{3,c} &= (\{03\} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \cdot s_{3,c}).
 \end{aligned}$$

Obrázek 2.2: Vzorce používané v kroku `mix_columns`

2.8 Operace `add_round_key`

Na každém prvku matice reprezentující jeden datový blok je v rámci operace `add_round_key` provedena operace XOR. Matice s daty má 16 prvků a část klíče, se kterou v této fázi pracujeme je také šestnáctiprvková. Funkce

XOR je tedy realizována vždy nad daným prvkem datové matice a odpovídající částí rozšířeného klíče. Když bychom tedy například uvažovali první prvek matice, mohl by pseudokód vypadat následovně:

```
1 prvek_matice[0] = prvek_matice[0] ^ round_key[0];
```

Kód 2.2: Princip operace `add_round_key`

3 Popis implementace

V rámci této kapitoly jsou popsány funkce obsažené v programu, jež se týkají algoritmu AES. Jedná se tedy o funkce obsažené v souboru `encrypt_logic.c`.

Program samozřejmě obsahuje i jiné funkce. Například jsou obsaženy části kódu umožňující načtení a zápis do souboru. Takové funkce jsou řádně okomentovány ve zdrojovém kódu a jsou ve většině případů triviální, jejich popis tedy nebude předmětem této kapitoly.

3.1 Použitý programovací jazyk, vývojové prostředí

Program jsem vytvořil v programovacím jazyce C a vývoj probíhal v rámci prostředí CLion.

Zmíněný jazyk jsem zvolil hlavně kvůli jeho nízkourovňosti a možnosti správy paměti programátorem. Zejména při šifrování větších souborů by rychlost programu implementovaného v C pravděpodobně předčila implementace v jiných, vysokoúrovňových jazycích (např. Java).

3.2 Funkce `key_expansion(unsigned char *original_key)`

Obsahuje implementaci operace popsané v rámci kapitoly 2.3.

Parametr `original_key` je pointer na původní, uživatelem volený, 16B klíč.

3.3 Funkce `key_expansion_core(unsigned char *sequence, unsigned char it_index)`

Jedná se o implementaci operace, která byla teoreticky popsána v rámci kapitoly 2.4.

Parametr `sequence` představuje sekvenci bytů (4B), která bude v rámci funkce upravována. Parametr `it_index` pak slouží pro specifikování pořadí volání dané funkce. Na základě pořadí volání pak probíhá výběr hodnoty ze substituční tabulky `r_con`, která je dostupná např. zde.

3.4 Funkce `sub_bytes(unsigned char *data_block)`

Implementace operace, jejíž popis je předmětem kapitoly 2.5.

Parametr `data_block` je 16-bytový datový blok, se kterým je zacházeno jako s maticí 4x4. Funkce zajišťuje nahrazení původních hodnot v matici hodnotou ze substituční tabulky, v kódu označena jako `s_box`.

3.5 Funkce `shift_rows(unsigned char *data_block)`

Daná operace je teoreticky popsána v kapitole 2.6.

V rámci kódu dojde k záměně jednotlivých prvků matice specifikované parametrem `data_block`, jež reprezentuje jeden 16B datový blok. Na začátku těla funkce tedy dojde k vytvoření pole s názvem `modifiedDataBlock`, do kterého jsou prvky postupně ukládány, aby nedošlo k přepsu dat původní matice. Poté, co jsou všechny prvky matice na požadovaných místech, dojde k přepsání hodnot v původní matici.

3.6 Funkce `mix_columns(unsigned char *data_block)`

Jedná se o implementaci operace, jejíž teoretický popis je předmětem kapitoly 2.7.

Vyhledávací tabulky zmiňované v teoretickém popisu jsou v kódu označeny jako `gal_mul2` a `gal_mul3`. Jednotlivé výpočty jsou průběžně ukládány do pole označeného jako `modifiedDataBlock`, aby během výpočtů nedošlo k nežádoucí úpravě dat v původní matici. Po dokončení výpočetních operací dojde k překopírování obsahu dat z pole `modifiedDataBlock` do původního pole, jež bylo do matice předáno.

3.7 Funkce `add_round_key(unsigned char *state, unsigned char *round_key)`

Implementace operace, jež byla teoreticky popsána v kapitole 2.8.

Parametr `state` specifikuje 16B datový blok, na kterém budou prováděny operace specifické pro krok `add_round_key`. Druhý parametr, `round_key`, specifikuje podklíč rozšířeného klíče, jež bude použit na operaci XOR.

3.8 Funkce `encrypt__state(unsigned char *data__block, unsigned char *expanded__key)`

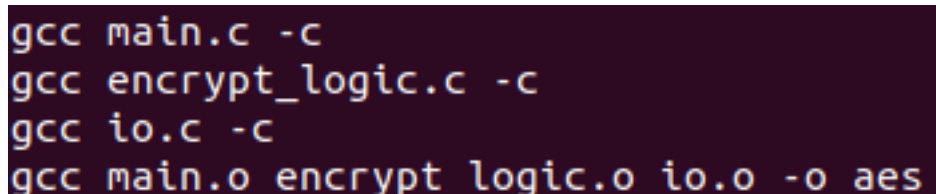
Aplikuje na 16-bytový datový blok posloupnost operací uvedenou v teoretické části (viz 2.1) a tím dojde k zašifrování daného bloku.

4 Uživatelská příručka

Program je možno spustit ve dvou režimech, umožňuje totiž šifrování uživatelem zadaného textu i celých souboru.

4.1 Překlad programu

Program obsahuje klasický makefile známý z Unixových operačních systémů. Pro překlad programu tedy stačí přepnout se v příkazové řádce (Terminálu) do složky s programem a zavolat příkaz **"make"**. V případě, že překlad proběhne v pořádku, uživatel spatří výpis analogický k textu na obrázku 4.1.



```
gcc main.c -c
gcc encrypt_logic.c -c
gcc io.c -c
gcc main.o encrypt_logic.o io.o -o aes
```

Obrázek 4.1: Úspěšný překlad programu

4.2 Spuštění v módu šifrování textu

Pro zašifrování textu musí být program spuštěn s následujícími parametry: `aes klic "'text_k_zasifrovani'"lokace_vystup`, kde:

- `aes` - název programu
- `klic` - klíč, pomocí kterého má být text zašifrován
- `text_k_zasifrovani` - text, jež má být zašifrován
- `lokace_vystup` - název souboru, do kterého bude výstup uložen NEBO - (pomlčka) pro výstup do konzole

4.3 Spuštění v módu šifrování souborů

Pro zašifrování souboru musí být program spuštěn s následujícími parametry: `aes klic soubor_k_zasifrovani lokace_vystup`, kde:

- `aes` - název programu
- `klic` - klíč, pomocí kterého má být soubor zašifrován
- `soubor_k_zasifrovani` - cesta k souboru, jež má být zašifrován
- `lokace_vystup` - název souboru, do kterého bude výstup uložen NEBO
- (pomlčka) pro výstup do konzole

5 Závěr

5.1 Funkčnost programu a jeho využitelnost v praxi

Program byl otestován na dodaných zkušebních souborech a poskytl očekávané výsledky. Mohu tedy prohlásit, že program pracuje dle očekávání.

Neočekávám využití programu dalšími lidmi, jelikož je na trhu dostupné množství zavedených a již léty prověřených alternativ. Aplikace také nedisponuje lákavým uživatelským rozhraním.

5.2 Problémy spojené s tvorbou programu

Program jsem nejprve zkoušel implementovat v programovacím jazyce Java, během pár hodin jsem však narazil na obtížnou práci s byty a program jsem raději přepsal do C. Hlavním problémem tedy byla zejména prvotní volba nevhodného vysokoúrovňového programovacího jazyka.

5.3 Zhodnocení, přínos úlohy pro řešitele

Program sice není prakticky využitelný, ale díky zadání jsem si vyzkoušel implementaci své první šifry a prohloubil si znalosti týkající se algoritmu AES.