Západočeská univerzita v Plzni
Fakulta aplikovanch věd
Katedra informatiky a výpočetní techniky

Bezpečnost v informačních technologiích

Advanced Encryption Standard

Plzeň, 2016 David Pivovar
celková doba řešení SP: 20 hodin

Obsah

Zadání

Kryptografie - implementace moderní šifry.

Implementace Advanced Encryption Standard pro šifrování a dešifrování dat v informačních technologiích.

1. Advanced Encryption Standard

Advanced Encryption Standard (AES) je standardizovaný algoritmus používaný k šifrování dat v informatice. Je založen na Rijndael agoritmu. Jedná se o symetrickou blokovou šifru šifrující i dešifrující stejným klíčem data rozdělená do bloků pevně dané délky.

AES používá k šifrování substituce i permutace. Pracuje s maticí bytů 4x4 označovanou jako stav. Má pevně danou velikost bloku na 128 bitů a velikost klíče na 128, 192 nebo 256 bitů. Velikost klíče určuje počet cyklů algoritmu.

- 128-bit klíč: 10 cyklů
- 192-bit klíč: 12 cyklů
- 256-bit klič: 14 cyklů

1.1 Popis algoritmu

1. Expanze klíče - podklíče jsou odvozeny z klíče pomocí seznamu Rijndael klíčů (Rijndael key schedule)
2. Inicializační část
   (a) Přidání podklíče - každý byte stavu je zkombinován s podklíčem za pomoci operace xor nad všemi bity
3. Iterace
   (a) Záměna bytů - každý byte je nahrazen jiným podle substituční tabulky
   (b) Prohození řádků - každý řádek stavu je postupně posunut o určitý počet kroků
   (c) Kombinování sloupců - zkombinuje čtyři byty v každém sloupci
   (d) Přidání podklíče

4. Závěrečná část
    (a) Záměna bytů
    (b) Prohození řádků
    (c) Přidání podklíče

2. Implementace

Program je napsána v jazyce C#.

Jedná se o konzolovou aplikaci. Na začátku se zvolí vstupní parametry, zadají se vstupní data. Výstup se na konci vypíše na konzoli a do souboru *result.dat*.

Program je rozdělen do tří tříd: inicializační, šifrovací, dešifrovací a třída se substitučními tabulkami.

2.1 Šifrování

Při šifrování zarovnáme zprávu na násobek 128 bitů a určíme velikost klíče a počet iterací. Vygenerujeme podklíče a postupně šifrujeme 128 bitové bloky.

2.1.1 Generování podklíčů

Nejprve se osmibitová slova klíče posunou doleva. Poté se nahradí dle substituční tabulky a zkombinují se operací XOR s odpovídajícími Rijndael klíčy.

2.1.2 Šifrování bloku

Ze 128 bitového bloku se vytvoří matice 4x4 po 8 bitech. Matice se zkombinuje operací XOR s odpovídajícím podklíčem. Následuje počet iterací dle délky klíče (klíč - 32 bity + 6). V iteraci se nahradí byty podle substituční tabulky, prohodí se řádky (posune byty v každé řádce o určitý offset doleva - první řádka je nezměněna, druhá o jedna, třetí o dva, čtvrtá o tři), zkombinují se 4 byty v každém sloupci (pomocí inverzní lineární transformace - každý sloupec je vynásoben s fixním polynomem) a přidá se podklíč. Na závěr se provede ještě jedna iterace bez kombinování sloupců.

Výsledná matice je zašifrovaný blok zprávy.

2.2 Dešifrování

Dešifrování probíhá podobně jako šifrování. Určíme velikost klíče a počet iterací a vygenerujeme podklíče. Následně postupně dešifrujeme jednotlivé 128 bitové bloky. Nazávěr odstraníme bílé znaky, kterými byla zpráva zarovnána na nasobek 128 bitů.

2.2.1 Dešifrování bloku

Dešifrování bloku probíhá ve stejném pořadí jako šifrování, pouze je prohozeno pořadí přidání podklíče a kombinace sloupců v iteraci. Záměna bytů se provádí podle inverzní substituční tabulky, v řádcích se posouvá doprava a sloupce se násobí inverzním polynomem než při šifrování.

Výsledná matice je dešifrovaný blok zprávy.

3. Uživatelská dokumentace

Po spuštění programu se zvolí mód: *e - šifrování* nebo *d - dešifrování*. Zvolí se vstup zprávy: *c - z konzole* nebo *f - ze souboru*. Zadá se zpráva, případně cesta k vstupnímu souboru, a klíč.

Program využívá kódování CP-1252. Toto kódování ignoruje většinu českých znaků.

Výsledný šifrovaný/dešifrovaný text se zobrazí na konzoli a uloží se do souboru *result.dat*

Závěr

Jazyk C# není pro šifrování zcela vhodný kvůli silné typové kontrole. Kvůli parsování zprávy na potřebné datové typy je také program závislý na kódování systému, na kterém byl přeložen.

Přesto algoritmus funguje rychle a spolehlivě i pro větší soubory.