Toteutusdokumentti

Aineopintojen harjoitustyö: Tietorakenteet ja algoritmit

2016 (Periodi II)

Markus Auvo

1 Ohjelman yleisrakenne

Ohjelman rakenne on varsin yksinkertainen. Ylimpänä on on pääluokka Main, jonka kautta algoritmitoteutuksia käytetään. Ohjelman varsinainen toteutus jakautuu itse algoritmit sisältävään pakkaukseen `algo` ja työkaluluokkia sisältävään pakkaukseen `util`. Testiluokilla on sama luokkajako.

2 Saavutetut aika- ja tilavaativuudet

Alla on esitetty pseudokoodina sekä salausavaimen muodostaminen että DES-algoritmin oma implementaatio. Algoritmissa ja sen salausavaimen luonnissa suoritetut permutaatio-operaatiot vaihtavat bittien paikkaa salattavassa bittilohkossa ennalta määritellyn taulukon mukaisesti. Tässä dokumentissa taulukoita ei ole tarpeen erikseen esitellä.

2.1 Salausavain

Salausavain on kiinteä 64-bittinen avain. Näin ollen avain koostuu tasan yhdestä lohkosta.

64-bittiseen lohkoon kohdistetaan aluksi permutaatio-operaatio *Permuted Choice 1*. Operaation tuloksena saadaan 56-bittinen lohko, joka jaetaan kahteen 28-bittiseen lohkoon.

Seuraavaksi suoritetaan 16 kertaa silmukka, minkä tuloksena saadaan 16 kappaletta salauksessa käytettäviä osa-avaimia. Yksi silmukan suoritus tuottaa siis yhden osa-avaimen.

Silmukan sisällä suoritetaan seuraavat operaatiot:

- Kumpaankin 28-bittiseen lohkoon kohdstetaan bittisiirto-operaatio, jossa lohkojen bittejä siirretään vasempaan.
- Lohkot yhdistetään takaisin 56-bittiseksi lohkoksi.
- 56-bittiseen lohkoon kohdistetaan permutaatio-operaatio *Permuted Choice 2*, minkä tuloksena saadaan 48-bittinen osa-avain.

Alla on yksinkertaistettu pseudokoodinen esitys salausavaimen muodostuksesta.

```
GENERATE_SUBKEYS(KEY)
    PERMUTATED_KEY = PERMUTED_CHOICE_1(KEY)
    C = PERMUTATED_KEY/2
    D = PERMUTATED_KEY/2
    FOR K=1 TO 16
        LEFT_SHIFT(C)
        RIGHT_SHIFT(D)
        CD = CONCATENATE(C,D)
        ROUND_SUBKEY[K] = PERMUTED_CHOICE_2(CD)
    END FOR
```

Jokainen silmukan sisällä suoritettava operaatio suoritetaan tasan kerran. Näin ollen jokaiselle silmukan sisäiselle käskylle saadaan vakio suoritusaika. Silmukan suorituskertojen määrä on myös vakio eli 16. Edellisestä seuraa, että salausavaimen muodostuksen aikavaativuus on vakio eli **Θ(1)**.

2.2 DES-algoritmi

Selväkielinen viesti on satunnaisen pituinen jono, joka jaetaan 64-bittisiin lohkoihin. Näin ollen viesti jaetaan *n* lohkoon. Viestin salauksen ja salauksen purkamiseen käytetään samaa salausavainta ja operaatioiden ero on järjestyksessä, jossa osa-avaimia käytetään. Viestin salauksessa käytetään osa-avaimia 1..16. Salauksen purkamiseen käytetään osa-avaimia 16..1.

Jokiseen lohkoon kohdistetaan ensin permutaatio-operaatio *Initial Permutation*. Tämän jälkeen suoritetaan 16 kertaa silmukka.

Silmukan sisällä suoritetaan seuraavat operaatiot:

- Salattava 64-bittinen lohko jaetaan kahteen 32-bittiseen osaan, joista oikean puoleinen puolilohko talletetaan tilapäismuuttujaan.
- Oikean puoleiseen puolilohkoon kohdistetaan laajennusrutiini *Expansion*, jossa puolilohkon kokoa kasvatetaan 48 bittiin.
- 48-bittinen lohko sekä 48-bittinen osa-avain annetaan syötteeksi XOR-operaatiolle.
- Tämän jälkeen XOR-operaation tuloksena saatu lohko 48-bittinen lohko jaetaan kahdeksaan 6-bittiseen osaan.
- Jokaiseen 6-bittiseen osaan kohdistetaan korvausoperaatio *Substitution*. Korvausoperaatioon käytetään kahdeksaa ennaltamääriteltyä taulukkoa, joita ei tässä dokumentissa erikseen esitellä.
- Korvausoperaation jälkeen oikean puolilohkon osiin kohdistetaan permutaatio-operaatio *Permutation*.
- Korvausoperaation jälkeen vasen ja oikea puolilohko annetaan syötteeksi XOR-operaatiolle, minkä jälkeen vasen puolilohko korvataan tilapäiseen muuttujaan tallennetulla, alkuperäisellä oikealla puolilohkolla.
- Puolilohkot yhdistetään takaisin 64-bittiseksi lohkoksi, johon lopuksi kohdistetaan vielä viimeinen permutaatio-operaatio *Final Permutation*, minkä tuloksena saadaan 64-bittinen salattu lohko.

Seuraavalla sivulla on yksinkertaistttu pseudokoodinen esitys viestin salauksesta.

```
CIPHER(MESSAGE)
    N = NUMBER_OF_MESSAGE_BLOCKS
    FOR K = 1 TO N
        MESSAGE_BLOCK[K] = INITIAL_PERMUTATION(MESSAGE_BLOCK[K])
        LEFT[K] = MESSAGE_BLOCK[K]/2
        RIGHT[K] = MESSAGE_BLOCK[K]/2
        FOR S = 1 TO 16
            TEMP_RIGHT = RIGHT[K]
            EXPANSION(RIGHT[K])
            EXCLUSIVE_OR(RIGHT[K], SUB_KEY[S])
            SUBSTITUTION(RIGHT[K])
            PERMUTATION(RIGHT[K])
            EXCLUSIVE_OR(RIGHT[K], LEFT[K])
            LEFT[K] = TEMP_RIGHT
            CIPHER_BLOCK = CONCATENATE(RIGHT[K], LEFT[K])
            FINAL_PERMUTATION(CIPHER_BLOCK)
        END FOR
    END FOR
```

Ulomman silmukan suorituskertojen määrä on täysin riippuvainen salattavan viestin pituudesta eli salattavien lohkojen määrästä *n*. Näin ollen ulompi silmukka suoritetaan *n* kertaa. Sisemmän silmukan suorituskertojen määrä on vakio eli 16. Edellisestä seuraa, että viestin salauksen aikavaativuus on *n*:stä riippuvainen eli **Θ(n)**.

3 Työn mahdolliset puutteet ja parannusehdotukset

Kun vertaillaan keskenään luokkia JavaDES ja MyDES, luokkien välinen ero suoritustehossa huomattava. Toisaalta, koska koko viesti salataan, algoritmia ei ole mahdollista toteuttaa logaritmisessa ajassa.

4 Lähteet

DES – Data Encryption Standard.
http://www.eventid.net/docs/desexample.asp