DSK Opgave 2
MosML DES

Kristian Thy
thy@itu.dk

---

1 Introduktion

Målet med denne opgave er at implementere funktioner i MosML til DES-kryptering. Det færdige system håndterer DES-kryptering i både ECB[1]- og CBC[2]-mode samt 3DES-kryptering.

2 Specifikation

Til DES-algoritmen skal bruges en række funktioner, som her kort beskrives sammen med de datatyper der forventes som input til og output fra disse. Jeg følger navnekonventionerne i [4].

Da ML ikke har nogen datatype for bits, vælger jeg at repræsentere bitstrenge som `int list`er indeholdende 0 og 1. Hvor der herunder nævnes "binære tal", er det denne repræsentation der hentydes til.

str2bin : string → int list

Konverterer en streng til det korresponderende binære tal.

bin2str : int list → string

Konverterer et binært tal til den korresponderende streng.

XOR : int × int → int

$\oplus$	0	1
0	0	1
1	1	0

[1] Electronic Code Book
[2] Cipher Block Chaining

xor : int list × int list → int list

XOR af to binære tal.

ls : α list × int → α list

Left shift n pladser på et binært tal.

keys : α list → α list list

Givet et binært tal på 64 bit, genererer `keys` de 16 korresponderende rundenøgler.

ip : α list → α list

Den initiale permutation, som bruges inden første runde.

ip' : α list → α list

Den initiale permutations inverse.

E : int list → int list

Ekspansions-permutationen, som bruges i `F`.

P : int list → int list

P-permutationen, som bruges i `F`.

F : int list × int list → int list

Den mystiske funktion der krypterer 32 bits. Input er de 32 bits der skal krypteres og rundenøglen på 48 bits.

ECBenc : string → string → string

ECB-kryptering af en streng givet et 8-bytes kodeord.

ECBdec : string → string → string

ECB-dekryptering af en streng givet et 8-bytes kodeord.

CBCenc : string → string → string

CBC-kryptering af en streng givet et 8-bytes kodeord.

CBCdec : string → string → string

CBC-dekryptering af en streng givet et 8-bytes kodeord.

TrDESenc[3] : (string × string × string) → string → string

3DES-kryptering af en streng givet tre 8-bytes kodeord.

TrDESdec : (string × string × string) → string → string

3DES-kryptering af en streng givet tre 8-bytes kodeord.

Hele signaturen kan ses i `http://www.itu.dk/people/thy/dsk/des/DES.sig` - de funktioner der er markeret "`(* hidden *)`" er dem, der ikke ville være tilgændelige hvis implementationen her skulle laves som et brugbart API, men for nemheds skyld har jeg gjort dem tilgængelige i denne test-version. Der kan da eksperimenteres direkte med de enkelte delfunktioner.

3 Implementation

Herunder beskrives implementationsspecifikke overvejelser. Der bruges ikke yderligere plads på de elementære substitutions- og permutationsfunktioner, da de bare er implementeret med en naiv $O(n^2)$ algoritme.

str2bin

Ved brug af `Int.fmt` der formaterer et heltal i base 10 til en streng-repræsentation af et heltal i base 2, konverteres en streng char for char til dens binære talrepræsentation.

bin2str

Regner et binært tal om til et heltal i base 10 og returner en streng med de korresponderende chars.

XOR

Simpel implementation af $\oplus$ ved hjælp af `case`-sætninger.

[3] Funktionsnavne i ML må desværre ikke starte med tal – `3DESenc` var ellers et oplagt navn.

xor

Tager rekursivt `XOR` på to lister, et element af gangen.

ls

Laver n rekursive 1-bits left shifts.

keys

Heri indgår `PC1` og `PC2`, permuted choice 1 og 2. Der er blot tale om simpel permutation i disse. Nøglerne genereres ved at bruge `PC1` på nøglen, og derefter bruge `PC2` på forskellige left shifts af `PC1`. Der returneres en liste af nøgler.

F

Input til `F` er den ekspanderede plaintext `E(p)` og rundenøglen `k`. Disse `xor`'es og der foretages et opslag i S-boksene på baggrund heraf. Dette lookup foregår som elegant vist i figur 3.9 i [4] ved at definere første og sjette bit som rækkenummeret og anden til femte bit som kolonnenummeret på en opdeling af `E(p) xor k` i grupper à 6 bits. Valget af matricer (datatypen `Array2`) til S-boksene skyldes at opslaget da kan foretages direkte på grundlag af en udregning af bitværdierne.

crypt

Krypteringsfunktion der ligger til grund for de højere-ordens krypteringsfunktioner. Første argument, `mode`, er enten "encrypt" eller "decrypt", alt efter den ønskede opførsel. Eneste forskel er at ved dekryptering reverseres rækkefølgen af rundenøglerne.

Ud over brugen af først den initiale permutation, et `swap` som jo egentlig blot er et 32-pladsers left shift, og til sidst den inverse permutation, domineres funktionen af de seksten rundekrypteringer, der krypterer halvdelen af input og swapper de to halvdele.

ECBenc

Padder input så længden af det er 0 modulo 8. Der paddes med EndOfText karakterer (ASCII kode 3), da disse kan formodes ikke at forekomme midt i en plaintext. Herefter krypteres 8-bytes blokke med `crypt` og konkateneres.

ECBdec

Dekrypterer 8-bytes blokke, og fjerner derefter evt. padding fra krypteringen.

CBCenc

Først genereres en initialiseringsvektor IV – et 64-bits binært tal. Herefter bruges samme algoritme som i `ECBenc`, på nær at forudgående blok `xor`'es med den aktuelle blok før kryptering. IV sendes med som første blok, så modtageren har en chance for at dekryptere beskeden.

CBCdec

Dekryptering i CBC-mode er simplere end kryptering. Der er kun et specialtilfælde, nemlig første blok som er den ECB-krypterede IV. Herefter dekrypteres blokkene rekursivt og `xor`'es med den forudgående blok.

TrDESenc og TrDESdec

Triviel funktionssammensætning. Virker kun med ECB-mode pt., formentlig på grund af designbeslutningen om at inkludere IV som første blok i CBC-mode. Kommentarer til hvordan dette løses modtages med kyshånd på `thy@itu.dk`.

4 Verifikation

For at verificere algoritmen har jeg benyttet [1] til at fremstille en test af krypterings- og dekrypteringsalgoritmerne[4]. Da verifikationen giver det forventede resultat, betragter jeg implementationen som korrekt (men ikke hurtig).

Litteratur

[1] Rivest, Ronald: *Testing Implementations of DES.*
`http://theory.lcs.mit.edu/~rivest/destest.txt`

[2] RSA Security: *Frequently Asked Questions about Today's Cryptography, v4.1.*
`http://www.rsasecurity.com/rsalabs/faq/`

[3] Sestoft, Peter: *MosML Library Reference.*
`http://www.dina.kvl.dk/~sestoft/mosmllib/`

[4] Stallings, William: *Cryptography and Network Security.* Tredje udgave, Prentice Hall, 2003

[4] `http://www.itu.dk/people/thy/dsk/des/Destest.sml`