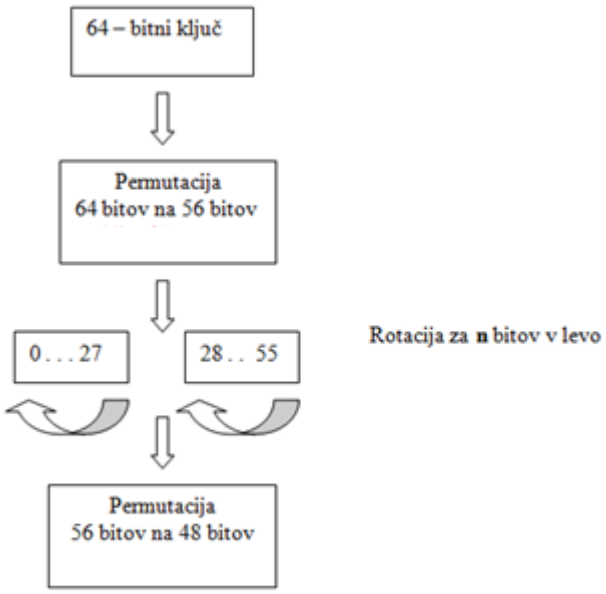
DES (Data Encryption Standard) šifriranje

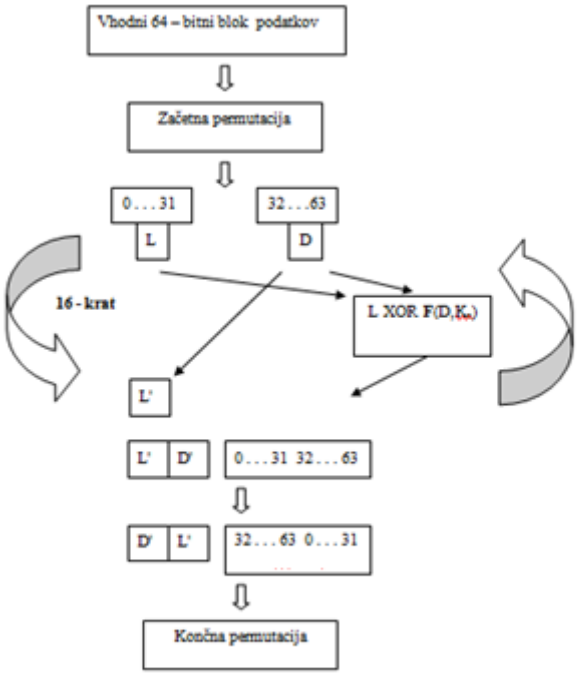
1. V programskem jeziku python napišite funkcijo **encrypt(vhodni_niz, kljuci)** za šifriranje bloka 64 bitov v skladu z DES algoritmom:
    a. Z uporabo funkcije **text2bin()** pretvorite vhodni niz znakov v blok 64 bitov.
    b. Pripravite ključe:
        a. Definirajte funkcijo **prepare_keys(key)**, ki ob vhodnem ključu v obliki besedila vrne seznam 16 zaporednih ključev v binarni obliki, ki jih algoritem potrebuje za enkripcijo. Način izdelave ključev je predstavljen spodaj:
            i. Pretvori besedni ključ v binarno obliko
            ii. Za začetno permutacijo 64 -> 56 bitov uporabite funkcijo **permute(kljuc, keyp, 56) –** pri tem je keyp tabela zamenjav (imate podano).
            iii. Za 16 zaporednih rotacij in permutacij 56 -> 48 bitov uporabite funkciji **shift_left(left, shift_table[i])** in **permute(levi_in_desni_kljuc_skupaj, key_comp, 48),** kjer je **,** key_comp podana tabela zamenjav, shift_table[i] pa podana tabela zamikov, ki jih uporabite na VSAKI izmed polovic ključa .
            iv. Podani tabeli zamenjav sta del DES algoritma.



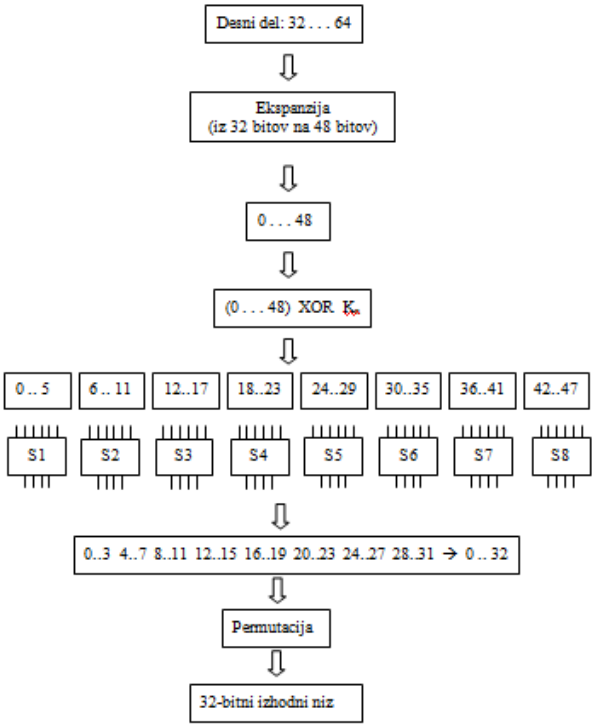
Slika 1: Priprava ključev.

    c. Znotraj šifrirne funkcije najprej kličite funkcijo, ki pripravi ključe.

d. Vhodno besedilo pretvorite v binarno obliko.
e. Prvo permutacijo bitov vhodnega niza opravite s funkcijo **permute(pt, initial_perm, 64).**
f. Razdelite blok bitov na levega in desnega.
g. 16 zaporednih jedrnih xOR operacij opravite s funkcijo **xor()**. Operacijo xor izvedite med izhodom funkcije F(desni_del_vhodnega_niza,trenutni_kljuc) in levim delom vhodnega niza. Po vsaki permutaciji (razen zadnji) je potrebno zamenjati levi in desni del bloka bitov.
h. Pravilno združite oba dela bloka v izhodni končni blok bitov.
i. Zadnjo permutacijo bitov vhodnega niza opravite s funkcijo **permute(zdruzeni_blok, final_perm, 64).**
j. Pretvorite končni blok bitov v niz znakov.



Slika 2: Jedro DES algoritma.



Slika 3: Funkcija F jedra DES algoritma.

2. Z namenom preizkusa delovanje funkcije *encrypt(vhodniNiz, kljuc)* šifrirajte besedo 'skrivnost' s ključem 'geslogeslo'.

3. Preverite koliko bitov šifropisa se spremeni s spremembo enega bita čistopisa. V ta namen z uporabo funkcije *encrypt(vhodniNiz, kljuc)* šifrirajte besedi 'cistopis' in 'Cistopis' s ključem '12345678' in primerjajte rezultata.

    a. Namig: število različnih bitov lahko preverite z funkcijo **diff_bits(bits1,bits2)**

4. Napišite funkcijo decrypt(sifropis,kljuc), ki dešifrira šifropis po algoritmu DES.

    a. Namig: Algoritem dešifriranja je enak šifriranju, samo vrstni red ključev je obraten.

5. Sestavite ECB (*Electronic Code Book*) modul za DES šifriranje bitov n_{biti}>64.
    a. Zakaj ta način ni najbolj varen?_____
6. Ugotovite vpliv dolžine ključa na varnost šifriranja. Dešifrirajte šifropis *v binarnem zapisu »111101110100001100111001011011110111001110000000001111010011101O«*. Slednji je šifriran v skladu s postopkom DES, prva 2 znaka uporabljenega ključa (v obliki besedila) sta neznana iz abecede malih tiskanih črk, ostali znaki so »0«.
7. Sestavite funkcijo fun_**3DES**(), ki bo opravila šifriranje ali dešifriranje po postopku Tripple-DES (3DES)

RSA algoritem

Je asimetrični šifrirni algoritem, ki uporablja par ključev, javni in zasebni.

- Pošiljatelj šifrira sporočilo z javnim ključem prejemnika (modularna eksponentna funkcija $s=c^e \ mod \ n$, ki se izvaja kot zaporedje modularnega množenja).
- Prejemnik dešifrira sporočilo z zasebnim tajnim ključem (modularna eksponentna funkcija $c= s^d \ mod \ n$, ki se izvaja kot zaporedje modularnega množenja).
- RSA izkorišča težavnost faktorizacije velikih števil.

Na osnovi znanega javnega ključa, čistopisa in šifropisa v realnem času ni mogoče ugotoviti zasebnega ključa. Dolžina ključa je odvisna on zahtevane stopnje varnosti.

RSA je počasnejši od blokovnih šifrirnih postopkov kot sta npr. 3DES ali AES. Zato je pogost naslednji šifrirni postopek med npr. Metko in Lukom:

- Metka šifrira sporočilo (čistopis) po AES postopku z naključno izbranim AES ključem.
- Metka zahteva Lukin javni RSA ključ s katerim šifrira uporabljen AES ključ. Oboje pošlje Luki.
- Luka najprej z lastnim zasebnim RSA ključem dešifrira Metkin AES ključ.
- S tem AES ključem nato dešifrira tudi vsebino sporočila, šifriranega s tem AES ključem.

Naloga:

1. Generirajte si par javnega in zasebnega ključa za RSA
   - Izberite dve praštevili *p* in *q (manjši od 100)*.
   - Izračunajte zmnožek *n = p * q.*
   - *Izračunajte število $\Phi = (p-1)*(q-1)$*
   - Določite število *e*, tako da je največji skupni delitelj števil *e* in Φ enak 1, kar pomeni, da sta si števili *e* in Φ tuji.
   - Določite število *d*, tako da velja da je *(e*d-1)* deljiv s *(p-1)*(q-1),*
     ali zapisano drugače: (*e*d*) mod Φ = 1.

Rešitev:

Javni ključ: e = _____ n = _____

Zasebni ključ: d = _____

Naloga:

2. Napišite funkcijo *fnRSA(besedilo,kljuc,n)* za šifriranje in dešifriranje v skladu z RSA algoritmom. Vhodni argument *kljuc* predstavlja število e oz. d, glede na smer šifriranja. Vsak znak besedila šifriramo posebej (pretvorimo ga v ASCII vrednost in obratno).
   - Čistopis (besedilo) šifrirajte po izrazu $s=c^e \ mod \ n.$
   - Šifropis (besedilo) dešifrirajte po izrazu $c=s^d \ mod \ n.$

3. Preizkusite delovanje funkcije *fnRSA*:

- Postavite vrednosti praštevil p=61 in q=53.
- Kolikšni sta vrednosti ključev e in d?
- Šifrirajte sporočilo 'Preizkus delovanja'.
- Dešifrirajte šifropis iz prejšnje točke.

Rešitev:

Ključa: e = _____ d = _____

Šifropis: _____

Dešifriran šifropis: _____

Napadi na RSA

V primeru, da sta uporabljeni praštevili p in q znani ali na preprost način določljivi, je zasebni ključ d določljiv na podlagi znanega javnega ključa e.

Pomanjkljivost majhnega ključa e: V takem primeru je dešifriranje kratkega sporočila ($c^e<n$) možno z iskanjem e-tega korena šifropisa, saj je $s=c^e$. Tak napad omogoča vpogled v vsebino šifriranega sporočila tudi brez poznavanja zasebnega ključa.

Preprosta metoda ugibanja vsebine sporočila in preverjanja pripadajočega šifropisa, saj je RSA deterministične šifrirni postopek:

- $s\_ugib=c\_ugib^e \bmod n$
- $s=c^e \bmod n$
- Ali sta *s_ugib* in *s* enaka?

Nevarna implementacija in hramba ključev.

Prestrezanje sporočil ('man in the middle').

Naloga:

4. Izvedite napad na šifropis shranjen v datoteki *sifropis-RSA.txt*, ki je šifriran z javnim ključem e=13, n=527.

Rešitev:

Pomoč:

Za odpiranje datoteke z UTF-8 znaki v pythonu lahko uporabite:

```
import codecs

file = codecs.open('sifropis-RSA.txt', 'r', 'utf-8')

sifropis = file.read()
```