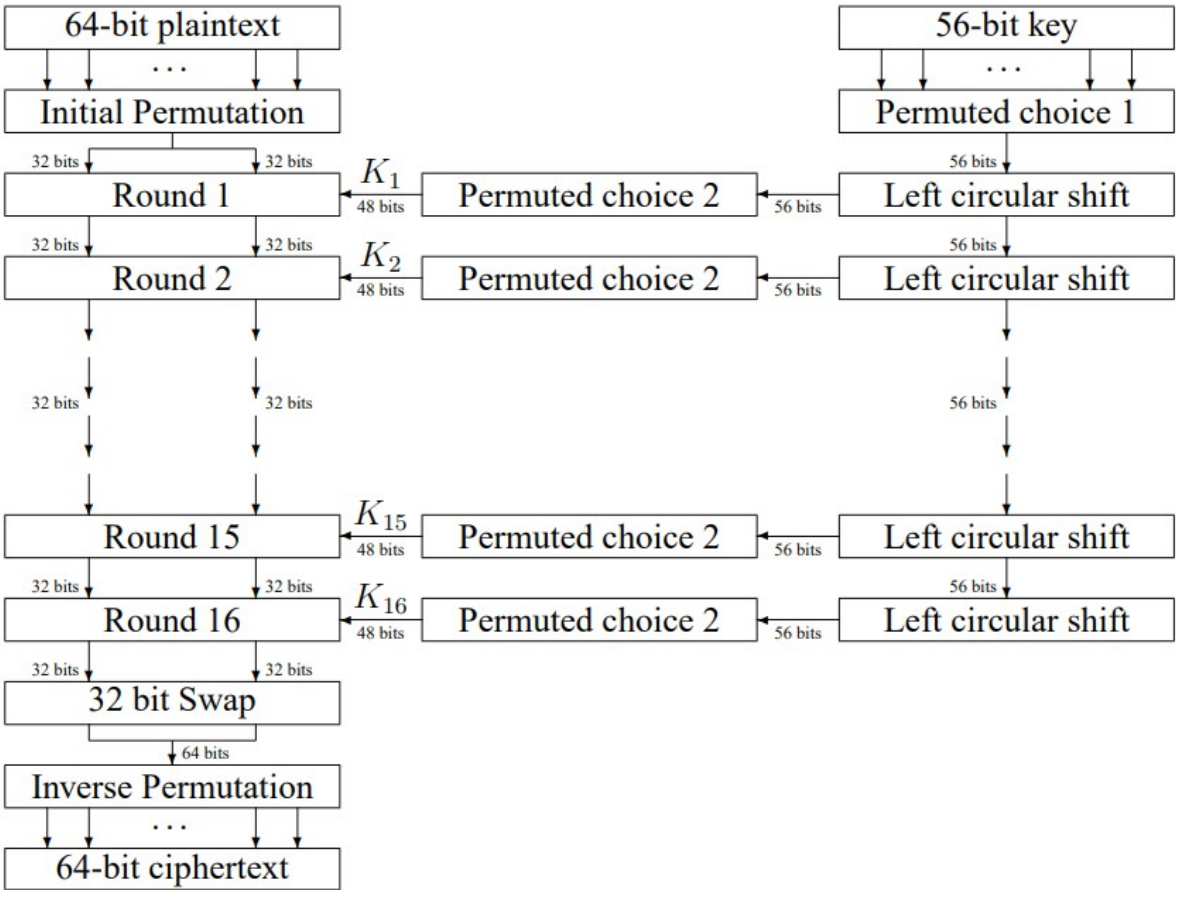
Úkol 2 - DES

Vytvořte v programovacím jazyce Python, C/C++ nebo Java program, který bude šifrovat a dešifrovat soubory prostřednictvím algoritmu *DES (Data Encryption Standard)*. DES je bloková šifra, která využívá bloky o velikosti 64 bitů. Cílem je načíst soubor jako pole bajtů, provést šifrování pomocí DES a výsledek následně uložit jako soubor `<nazev_souboru>_<pripona>.des`

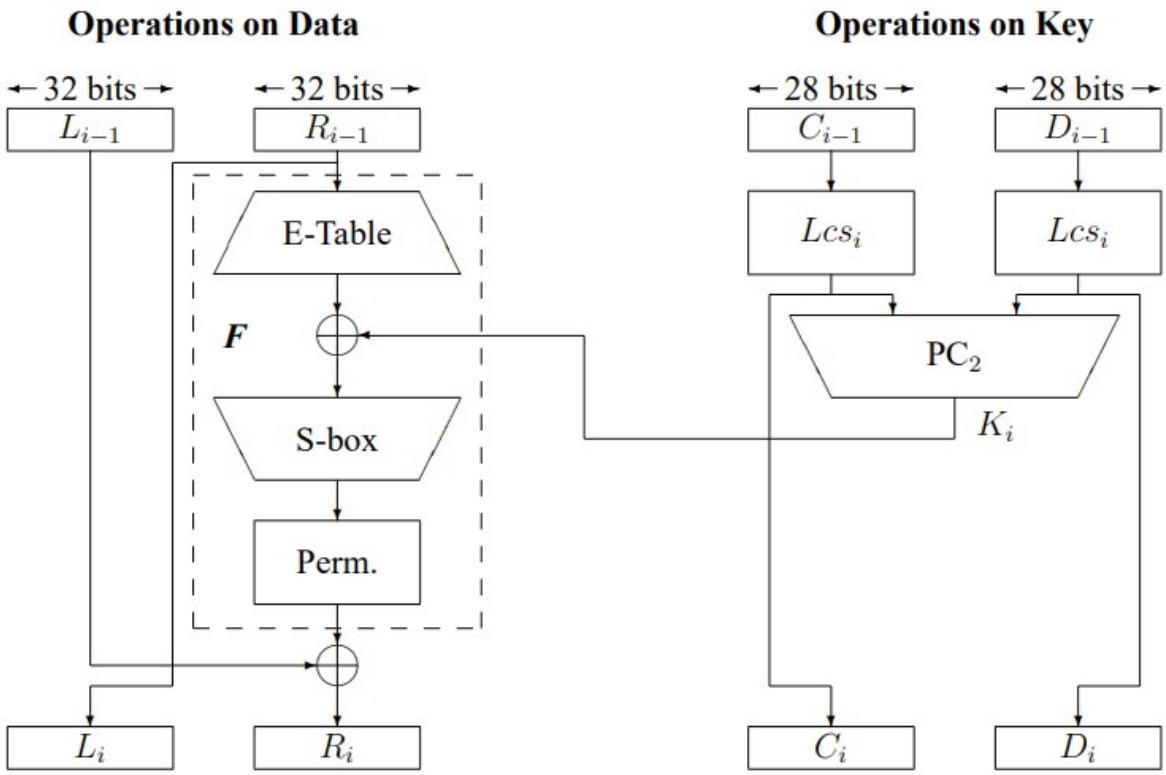
Pro implementaci zvolte nejjednodušší mód provozu, tzn. *Electronic Code Book (ECB)*. Budete tedy šifrovat blok po bloku nezávisle na sobě, dokud nezpracujete celý vstupní soubor.

Pseudonáhodný 64-bitový DES klíč si vygenerujte a uložte do souboru **`key.txt`** v hexadecimálním formátu (např. **b97ed418e9733e53**). Pomocí tohoto vygenerovaného klíče zašifrujete postupně všechny soubory ve složce **`validation/`**

Na obrázku 1 je schéma šifrování. Je potřeba provést redukci klíče na 56 bitů (viz popis algoritmu [1] nebo [1]). Vzhledem k tomu, že DES je symetrický šifrovací algoritmus, tak schéma dešifrování je naprosto totožné s tím rozdílem, že jednotlivé části klíče se použijí v opačném pořadí (tzn. při šifrování K_1 - K_{16} a při dešifrování K_{16} - K_1). Algoritmus se skládá z 16 iterací (anglicky *rounds*), které jsou identické, pouze se při nich mění parametr – podklíč o délce 48 bitů pro danou iteraci (viz schémata na obr. 1 a 2).



Obrázek 1: Diagram DES pro šifrování, zdroj: http://www.facweb.iitkgp.ac.in/~sourav/DES.pdf



Obrázek 2: Detailní schéma jedné iterace, Feistelova síť, zdroj: http://www.facweb.iitkgp.ac.in/~sourav/DES.pdf

Potřebné tabulky, S-boxy a P-boxy a tabulky si vytvořte dle specifikace DES. Není třeba je ukládat do speciálních souborů, postačí je mít uvedené ve zdrojovém kódu.

Vzhledem k tomu, že úkolem je (de)šifrovat libovolný soubor, je zapotřebí vyřešit problém, kdy velikost vstupního souboru nebude dělitelná 64 bity (neboli jinými slovy poslední blok bude kratší než 64 bitů). K tomu využijeme tzv. *nulový padding* – doplnění nulovými bity tak, aby se zarovnala velikost souboru do požadované délky.

Implementujte nulový padding následujícím způsobem:

1. Nejprve zjistěte, kolik nulových bajtů je potřeba doplnit (v našem případě 8-bajtových bloků je to buď 1, 2, 3, … nebo 7).
2. Vygenerujte potřebný počet nulových bajtů a doplňte je do načteného pole bajtů před spuštěním algoritmu DES.
3. Při dešifrování a vytváření původního souboru je nutné nulový padding odstranit a je zapotřebí vědět, kolik bajtů bylo takto “uměle” přidáno. Proto do posledního bajtu paddingu zakódujte celkový počet “paddovaných” bajtů.
4. Po dešifrovacím procesu DES si přečtěte poslední bajt, odstraňte správný počet přidaných bajtů a zbývající data potom uložte do souboru (z názvu zašifrovaného souboru si dekódujte původní název souboru a jeho příponu).

Pozor! Pokud je velikost souboru dělitelná velikostí bloku může se zdát, že nemusíme přidávat žádný padding. Vzniká ale otázka, jakým způsobem se dozvíme, zda dešifrovaná sekvence bajtů byla původně "paddována" či nikoliv? Tuto situaci musíme ošetřit. Budeme postupovat tak, že poslední bajt bude **vždy a za každých okolností** obsahovat počet přidaných bajtů. Z toho vyplývá, že v případě souboru, jehož velikost je dělitelná velikostí bloku přidáme rovnou **8 "nových bajtů"** s tím, že v posledním bude uvedeno zakódováno číslo 8, protože 8 bajtů (64 bitů) bylo přidáno.

Příklad 1:
Soubor **a.txt** má velikost **80 B**, obsahuje tedy 10 bloků o délce 64 bitů. Před šifrováním přidáme následující padding (znak '|' je zde jenom jako oddělovač z důvodu přehlednosti):
`00000000|00000000|00000000|00000000|00000000|00000000|00000000|00001000`
neboli hexadecimálně
`0x0000000000000008`

Příklad 2:
Soubor **b.txt** má velikost **81 B**, obsahuje tedy 10 bloků o délce 64 bitů + další jeden blok, který má pouze 8 bitů → je nutné provést zarovnání na 11 bloků, každý o délce 64 bitů. Před šifrováním přidáme následující padding (znak '|' je zde jenom jako oddělovač z důvodu přehlednosti):
`00000000|00000000|00000000|00000000|00000000|00000000|00000111`
neboli hexadecimálně
`0x00000000000007`
Podobně i pro další případy, které mohou nastat.

Berte v úvahu i možnost, že testování může probíhat na poměrně velkých souborech. Váš program musí být tedy dostatečně efektivní, aby doběhl v rozumném čase. Při běhu programu proto informujte uživatele o počtu zpracovaných bloků (např. po zpracování každých 10 % bloků vypište: (**Zpracováno: current_block / all_blocks**).
Případně implementujte nějaký progress bar (v Pythonu např. TQDM a jiné).

Součástí archivu se zadáním je následující:

- **`main.py`**
- **`des.py`**
- **`test/`**
- **`out/`**
- **`decoded/`**
- **`validation/`**

Soubory **main.py a des.py** je kostra, kterou můžete použít nebo ze které můžete vyjít; můžete je ale i ignorovat a program si zcela vytvořit dle sebe (např. dekomponovat jiným způsobem). Program nicméně bude očekávat jediný argument a to buď **-e** nebo **-d** a dle toho se rozhodne, zda se bude šifrovat nebo dešifrovat. Žádný další argument nebude vyžadován. V případě šifrování

bude úkolem programu v cyklu projít všechny soubory ve složce **`validation/`** a postupně je zašifrovat do složky **`out/.`** V případě dešifrování program načte klíč ze souboru **`key.txt`,** projde všechny soubory ve složce **`out/`** a dešifruje je do složky **`decoded/`**

Ve složce **`validation/`** jsou validační soubory, na kterých vaši práci odladťe. Pro účely validace vaší práce jsou k dispozici ve složce **`test/`** jednotkové testy. Silně doporučujeme používat testovací výpisy po jednotlivých krocích a jejich porovnání s referenční implementací (viz doporučená literatura [2]). V kostře máte také připravený “debug” vstup, na kterém vaši implementaci odladíte (přesně dle [2]). Doporučujeme pro každou prováděnou operaci (permutace, dělení bloku na left a right, substituce atd.) vytvořit funkce. Tyto dílčí funkce si odladťe, aby vracely správný požadovaný výstup a až pak stavte jednotlivé “stavební bloky” za sebe až vznikne DES.

Na konci šifrování vám musí vyjít přesně to, co je uvedeno v [2]. Padding v rámci “debug” módu neřešte. Po odladění přepněte flag “debug” na **False** a implementaci rozšiřte na zpracování souborů ve složce **`validation/`** (tzn. včetně paddingu popsaného výše).

Příklad výstupu

Uvažujme dva soubory ve složce **`validation/`** např. **`validation/dwarf.bmp`, `validation/Applied_Cryptography_(Bruce Schneier).pdf`**
Váš program po spuštění s argumentem **-e** (**`python main.py -e`**) vytvoří zašifrované soubory **`out/dwarf_bmp.des`** a **`out/Applied_Cryptography_(Bruce Schneier)_bmp.des`**

Pustíme-li následně program s parametrem **-d** dojde k dešifrování souborů **`out/dwarf_bmp.des`** a **`out/Applied_Cryptography_(Bruce Schneier)_bmp.des`** a výsledkem budou soubory **`decoded/dwarf.bmp`, `decoded/Applied_Cryptography_(Bruce Schneier).pdf`**
Soubory ve složce **`decoded/`** a **`validation/`** musí být naprosto totožné a nepoškozené. Toto je jeden z testů, které se budou provádět.

Jednotkové testy

Součástí archivu se zadáním je i složka **`test/`** a v ní skript s jednotkovými testy v Pythonu. Pokud implementujete v jiném jazyku než v Pythonu, doporučujeme si sadu podobných jednotkových testů také napsat.

Testy porovnávají a testují:

1. správnou velikost a formát vygenerovaného klíče
2. kontrolu paddingu
3. zda zpětně dešifrovaný soubor se shoduje s původními (resp. zda obsah složek **validation/** a **decoded/** je stejný).

Odevzdání

Vytvořte odevzdávací archiv, který pojmenujete **`<vaše_osobni_cislo>.zip`** a zabalte do něj následující:

- vaše spustitelné programy včetně zdrojových kódů
- složku **`validation/`** s validačními soubory a prázdné složky **`out/`** a **`decoded/`**

K zamyšlení

Při implementaci si uvědomte, jakým způsobem se pracuje s klíčem a jak je transformován na jednotlivé podklíče. Najděte si pak v literatuře výčet a příklady tzv. slabých klíčů a odpovězte si na otázku, proč jsou vlastně slabé a jaký dopad to bude mít na algoritmus DES.

Doporučená literatura

[1] **Applied Cryptography –** Chapter 12 Data Encryption Standard (DES)

[2] **https://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm**