Úkol 2 - AES [8 bodů]

Vytvořte v programovacím jazyce Python, C/C++ nebo Java program, který bude šifrovat a dešifrovat soubory prostřednictvím algoritmu AES (Advanced Encryption Standard). Uvažujme délku bloku 128 bitů. Cílem je načíst soubor jako pole bajtů, provést šifrování a výsledek následně uložit jako soubor **`<nazev_souboru>.aes`**

Pro základní implementaci zvolte nejjednodušší mód provozu, tzn. Electronic Code Book (ECB). Budete tedy šifrovat blok po bloku nezávisle na sobě, dokud nezpracujete celý vstupní soubor. Cílem práce je zašifrovat oba soubory (**`customers-20.csv`** a **`customers-2000000.csv`**) ve složce **`validation/`**

Můžete počítat s tím, že validační soubory (a i testovací) budou zarovnány na 16 bajtů a že všechny mají koncovku **`.csv`**. Není potřeba tak řešit situaci, kdy poslední 128-bitový blok bude neúplný. Jeden z validačních souborů je poměrně velký (cca **180 MB**), váš program musí být tedy dostatečně efektivní, aby doběhl v rozumném čase. Je proto **!!nutné!!**, abyste při běhu programu informovali 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:

- **`validation/`**

Součástí řešení bude samozřejmě i dešifrování a uvedení souboru do původního stavu. Program bude očekávat dva argumenty: parametr **-e** nebo **-d**, dle kterého se rozhodne, zda se bude soubor šifrovat nebo dešifrovat a název souboru. Žádný další argument nebude vyžadován. Vygenerujte si pseudonáhodný 128-bitový klíč a uložte ho v **hexadecimálním formátu** do souboru **`aes_key.txt`**.

Z dostupných zdrojů si nastudujte algoritmus a doporučujeme používat testovací výpisy po jednotlivých krocích a jejich porovnání s referenční implementací (viz např. [1])

Příklad výstupu

Pro soubor **`validation/customers-20.csv`** spustíte váš program následovně:

`python aes.py -e validation/customers-20.csv`
vytvoří se soubory **`aes_key.txt`** a **`customers-20.aes`**

Poté můžu spustit program takto:
`python aes.py -d customers-20.aes`
načte se soubor s klíči a vytvoří se soubor **`customers-20.csv`**
Je nutné, aby nově vytvořený dešifrovaný soubor byl nepoškozený a totožný s původním souborem ve složce **`validation/`**

Odevzdání

Vytvořte archiv **`<vaše_osobni_cislo>.zip`** a zabalte do
něj následující:

- vaše spustitelné programy včetně zdrojových kódů
- soubor **`README.txt`**

Do souboru **`README.txt`** stručně popište vaši práci. Složku **`validation/`** do vašeho odevzdávacího archivu nepřidávejte. Za takto vypracovanou práci získáte 8 bodů. Další 2 body lze získat vyřešením druhé části.

AES módy provozu CBC, CFB [2 body]

Implementujte do svého řešení módy provozu: Cipher Block Chaining (CBC) a Cipher FeedBack (CFB). Vstup a výstup bude totožný. Inicializační vektor o správné velikosti si vytvořte a mějte ho jako konstantu v kódu (nemusíte ho ukládat do souboru). Spuštění bude stejné jako v předchozí části s tím, že se vytvoří soubory do jejichž názvů přidáte "**`_cbc`**" nebo "**`_cfb`**" (tedy např. takto: **`customers-20_cbc.aes`** a **`customers-20_cfb.aes`**
Další parametr do svého programu nepřidávejte.
Pokud implementujete všechno, očekáváné chování je následující.
`python aes.py -e validation/customers-20.csv`
vytvoří se soubory **`aes_key.txt`**, **`customers-20.aes`**, **`customers-20_cbc.aes`** a **`customers-20_cfb.aes`**

Dešifrování:
`python aes.py -d customers-20.aes`
`python aes.py -d customers-20_cbc.aes`
`python aes.py -d customers-20_cfb.aes`
pokaždé se vytvoří znovu soubor **`customers-20.csv`**, který musí být totožný s původním souborem.

[1] https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf