Úkol 3 - Zavazadlový algoritmus

Vytvořte program, který bude provádět asymetrické šifrování, založené na principu zavazadlového algoritmu.
Program bude očekávat jeden parametr. Při zadání parametru **-e** se zašifrují všechny soubory ve složce **`validation/.`** Program si nejprve vygeneruje parametry ***p***, ***q*** a **privátní+veřejný klíč**. Vše se uloží do souborů v **dekadické** podobě:

- **`p.txt`**
- **`q.txt`**
- **`private_key.txt`**
- **`public_key.txt`**

Formát souborů s klíči bude sekvence dekadických čísel oddělených čárkou, např.: `51,78,198,619,1111,3255,7596,13533, …`
Velikost privátního a veřejného klíče **pevně nastavte na 250** (počet předmětů ve virtuálním zavazadle). Čísla (hmotnosti předmětů) volte tak, aby byla jejich délka (v bitech) v intervalu **100 až 400 bitů.** Parametr ***q*** bude mít aspoň **100 až 200 bitů**. U všech parametrů zkontrolujte validitu (tj. nesoudělnost parametrů, superrostoucí posloupnost, atd. detailněji viz studijní materiály).

Pro dešifrování budete potřebovat najít multiplikativní inverzi $\mathbf{p}^{-1}$. Pozor, multiplikativní inverze nemusí vždy existovat (viz studijní materiály). V takovém případě si přegenerujte všechny parametry tak, abyste našli multiplikativní inverzi $\mathbf{p}^{-1}$.
Vstupní soubor rozdělíte na bloky o velikosti počtu prvků klíče (tj. 250 bitů). Poslední blok zarovnejte zprava nulami (zero padding), bude-li to nutné. Každý blok dle algoritmu zašifrujete a výsledné číslo reprezentující blok bude uloženo ve výstupním textovém souboru (v dekadické podobě jako jeden řádek). Výstupní soubor tedy bude vypadat např. takto (viz Obrázek 1).

```
1  1527375732277848533153886032679263662561685407520201991792038 6
2  1732997102453494701143530691396159348669444090986878286839181 9
3  3485346801590276283580545763580441991977088576127547697056113 2
4  2370375370323172331411442246730886147381473520616982292610446 4
5  2539056914847673395078356735273349500048081101271831611640067 0
6  5382431644737775145132462242401253800136309950469685417301461 1
7  5165076791989686642040025409413165931663834500850305552948562 2
8  4608505970514348294025474786560243279061540209106303101385505 8
9  4987657431862509873612553876474277252377041806792180680170181 9
10 4582007640262715014107593082165904623423504949956003365104686 3
11 3961102309990392787925158432958005461733003726896783975643572 3
12 4362996449112665558757059461610950433942468740204238668675510 3
13 3835447851022305310419929417148710762052433397324361235037567 7
14 4994505822084790078644511499007986500051143198439537443558571 8
15 3844885910984866018616972650023566383315882766065404324086767 8
16 2962712864271167824166652830591460945066712633624425216858809 2
```

Obrázek 1: Příklad zašifrovaných bloků v souboru

Do názvu výstupního souboru si přidejte na začátek velikost paddingu (tzn. počet nulových bitů, které jste přidávali). Výsledný název souboru, který uložíte do složky **`out/`** bude **`<velikost_paddingu>_<filename>_<koncovka>.kna`**

Počet řádků v souboru se musí shodovat s počtem bloků. Každé číslo symbolizuje výslednou hmotnost zavazadla. **Pozor!** Nevytvářejte žádný binární soubor, musí se jednat o čitelný formát, který lze otevřít libovolným textovým editorem.

Po zadání parametru **-d** váš program provede dešifrování, tzn. projde všechny zašifrované soubory s koncovkou **.kna** ve složce **out/** a vytvoří z nich původní soubory, které uloží do složky **decoded/**. Z názvu souboru **.kna** si všechna potřebná metadata dekódujte.

Řešení je zcela ve vaší kompetenci. Pokud pro implementaci zvolíte Python, lze využít následující konstrukce pro generování náhodných čísel o určité velikosti *n* (v bitech):

```
import random
cislo = random.getrandbits(n)
```

Při programování nejprve doporučujeme algoritmus odladit na klíčích a parametrech malé velikosti (např. příklady z přednášek nebo cvičení) a až pak vygenerovat parametry zavazadlového algoritmu dle požadavků výše. Kontrolovat se bude validnost všech parametrů algoritmu a to, že dešifrované soubory budou nepoškozené a stejné jako původní soubory ve složce **`validation/`**

Jednotkové testy

Nejsou součástí archivu se zadáním.

Doporučená literatura

Applied Cryptography – 119.2 Knapsack Algorithms