Úkol 3 - Zavazadlový algoritmus

Vytvořte program, který bude jednoduchým způsobem provádět asymetrické šifrování, založené na principu zavazadlového algoritmu.

Součástí archivu se zadáním je následující:

- **`out/`**
- **`decoded/`**
- **`validation/`**

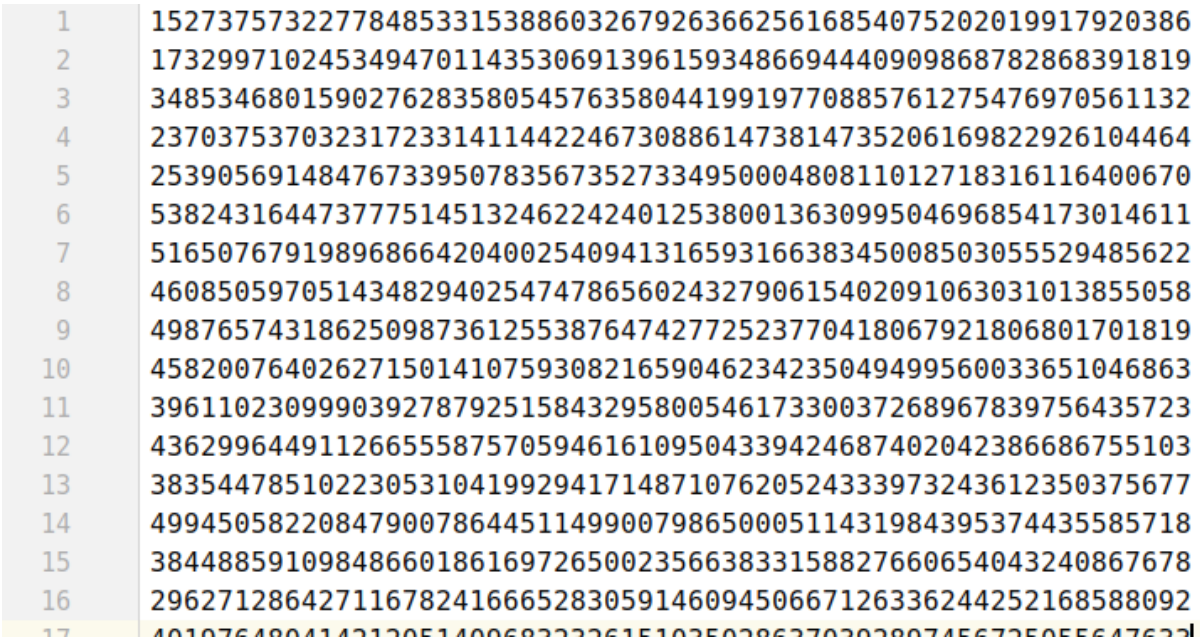
Program si nejprve vygeneruje parametry ***p***, ***q*** a **privátní+veřejný klíč**. Vše uložte do souborů v dekadické podobě:

- **`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 $\mathbf{p^{-1}}$. K tomu doporučujeme implementovat rozšířený Eukleidův algoritmus. Pozor, multiplikativní inverze nemusí vždy existovat (viz studijní materiály). V takovém případě si přegenerujte všechny parametry tak, abyste našli multiplikativní inverzi $\mathbf{p^{-1}}$.

Zašifrujte všechny soubory ve složce **`validation/.`** Začněte tím, že vstupní soubor rozdělíte na bloky o velikosti počtu prvků klíče (tj. 250 bitů). Poslední blok zarovnejte zprava nulami, bude-li to nutné. Poté každý blok zašifrujte a uložte ho v dekadické podobě jako nový řádek do textového souboru (**`*___encrypted___output.txt)`**. Dle algoritmu bude každému bloku přiřazeno číslo, které značí celkovou hmotnost předmětů v zavazadle. Výstupní soubor tedy bude vypadat např. následovně (viz Obrázek 1).



Obrázek 1: Příklad zašifrovaných bloků v souboru **`dwarf___encrypted___output.txt`**

Počet řádků v souboru se musí shodovat s počtem bloků. Každé číslo symbolizuje výslednou hmotnost zavazadla. Váš program přirozeně musí umět i dešifrování, tedy musí být schopen po načtení tohoto zašifrovaného souboru a znalosti privátního klíče vygenerovat původní bity (resp. bajty) souboru. Vytvořte znovu původní soubor a uložte ho do složky **`decoded/`**. Odstraňte nulové bity, které jste přidávali, bude-li to nutné.
Podobně jako v předchozí práci, dekódované bajty uložte v hexadecimální podobě do souboru **`out/*___decrypted___hexoutput.txt.`** Testovací soubory mohou být opět velké, uložte tedy jen prvních 100 bajtů – výsledný soubor tedy bude obsahovat pouze 1 řádek. Srovnáme-li prvních 100 bajtů původního souboru, musí se shodovat s obsahem souboru **`out/*___decrypted___hexoutput.txt.`**

Vyhodnocení bude probíhat automatickým validátorem na testovací sadě náhodně vybraných souborů. 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.

Příklad výstupu

Př. 1 Uvažujme jediný soubor např. **`validation/dwarf.bmp`**
Fáze generování parametrů:

a) vytvoření souborů: **`p.txt, q.txt, private_key.txt, public_key.txt`**
b) vytvoření multiplikativní inverze $\mathbf{p}^{-1}$
(pokud $\mathbf{p}^{-1}$ neexistuje → přegenerovat parametry)

Váš program následně vytvoří:

a) zašifrovaný soubor **`out/dwarf___encrypted___output.txt`**, kde bude seznam zašifrovaných bloků (viz Obrázek 1)

b) textový soubor **`out/dwarf___decrypted___hexoutput.txt`** a v něm jediný řádek obsahující prvních 100 dešifrovaných bajtů
c) dekódovaný soubor **`decoded/dwarf.bmp`**

Jednotkové testy

Nejsou součástí archivu se zadáním.

Doporučená literatura

Applied Cryptography – 119.2 Knapsack Algorithms