

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 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 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).

1	15273757322778485331538860326792636625616854075202019917920386
2	17329971024534947011435306913961593486694440909868782868391819
3	34853468015902762835805457635804419919770885761275476970561132
4	23703753703231723314114422467308861473814735206169822926104464
5	25390569148476733950783567352733495000480811012718316116400670
6	53824316447377751451324622424012538001363099504696854173014611
7	51650767919896866420400254094131659316638345008503055529485622
8	46085059705143482940254747865602432790615402091063031013855058
9	49876574318625098736125538764742772523770418067921806801701819
10	45820076402627150141075930821659046234235049499560033651046863
11	39611023099903927879251584329580054617330037268967839756435723
12	43629964491126655587570594616109504339424687402042386686755103
13	38354478510223053104199294171487107620524333973243612350375677
14	49945058220847900786445114990079865000511431984395374435585718
15	38448859109848660186169726500235663833158827660654043240867678
16	29627128642711678241666528305914609450667126336244252168588092
17	401076480414313051400683333615103503863703038074567350556476331

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:

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

Váš program následně vytvoří:

- 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) dekodovaný soubor `decoded/dwarf.bmp`

Jednotkové testy

Nejsou součástí archivu se zadáním.

Doporučená literatura

Applied Cryptography – 119.2 Knapsack Algorithms