BIT 2024/2025

Úkol 3 - Asymetrická kryptografie

RSA

Vytvořte v programovacím jazyce Python, C/C++ nebo Java program, který bude šifrovat a dešifrovat soubory prostřednictvím algoritmu RSA (Rivest, Shamir, Adleman). Vytvořte si soukromý a veřejný klíč - všechny parametry algoritmu (tj. čísla *p* a *q*, *e* a *n*) si vygenerujte a uložte do souboru v hexadecimálním formátu. Nezapomeňte na to, že musí existovat multiplikativní inverze čísla *e* (d nebo také e^{-1}). Zajistěte, aby parametry byly dostatečně velká prvočísla (např. tak, aby délka odpovídala doporučeným 2048 bitů).

Prvky privátního klíče uložte do souboru `priv_key.txt`. Veřejný klíč do souboru `pub_key.txt` v tomto formátu:

```
e=<hexa_string_cislo_e>
n=<hexa_string_cislo_n>     //soubor pub_key.txt

d=<hexa_string_cislo_d>
n=<hexa_string_cislo_d>     //soubor priv_key.txt
```

Cílem je poté načíst libovolný soubor jako pole bajtů, provést šifrování a výsledek následně uložit jako soubor `<nazev_souboru>_<koncovka>.rsa`

Program bude očekávat parametr `-e` nebo `-d`. Při šifrování se použije veřejný klíč a pro dešifrování klíč soukromý. Při načítání zašifrovaného souboru musíte vědět, jaký počet bajtů ze souboru musíte načíst, abyste byli schopni vytvořit původní blok souboru nezašifrovaného. Toto lze vyřešit například tak, že při šifrování budete vytvářet bloky o fixní velikosti (pochopitelně musíme přihlédnout k číslu *n*).

Příklad:
Zvolím si velikost bloku např. 256 bitů (padding není potřeba řešit, předpokládejte, že velikost souboru je dělitelná 256 bity). Na toto 256bitové číslo (32 bajtů) aplikuji šifrování RSA a zajistím, že výstupem je pokaždé číslo, které má např. 2048 bitů (bez ohledu na to, jak je velké). Tzn. v případě, že je výsledkem operace RSA číslo 3, výstupní blok bude tvořit 2046 nul a 2 jedničky na konci. Při dešifrování budu vědět, že mám načítat bloky o velikosti 2048.

Opět implementujte jednoduchý progress bar, který uživatele informuje o průběhu šifrování a dešifrování.

Příklad výstupu

Můžete použít soubor, který již máte z minulé samostatné práce **`validation/customers-20.csv`**

`python rsa.py -e validation/customers-20.csv`
vytvoří se soubory **`priv_key.txt`**, **`pub_key.txt`** a **`customers-20_csv.rsa`**

Poté můžu spustit program takto:
`python rsa.py -d customers-20_csv.rsa`
načte se soubor s klíči, z názvu souboru se dekóduje přípona 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. V **`README.txt`** zhodnoťte vlastnosti (výhody a nevýhody) implementovaného řešení. Za úspěšné vyřešení úlohy získáte 10 bodů.