Fakulta informačních technologií Vysokého učení technického v Brně

Projekt č. 2 - Implementace a prolomení RSA

Kryptografie – KRY

Adam Švenk (xsvenk00)

Úvod

Cieľom tejto dokumentácie je popísať implementáciu projektu č. 2 v predmete Kryptografia (KRY). Predmetom tohto projektu je implementovať asymetrický šifrovací algoritmus RSA v programovacom jazyku C/C++. Algoritmus RSA je asymetrický šifrovací algoritmus s verejným kľúčom. Funguje ako bloková šifra, kde blok je celé číslo medzi 0 a n. Úlohou programu je vygenerovať parametre RSA – verejný a súkromný kľúč, šifrovať a dešifrovať správy pomocou algoritmu RSA a prelomiť algoritmus RSA pomocou metódy faktorizácie slabého verejného modulu.

Generovanie parametrov

Generovanie parametrov algoritmu RSA, teda verejného a súkromného kľúča, predstavuje prvú časť implementácie. V celom programe je na prácu s veľkými číslami použitá knižnica `GMP`. Generovanie je možné spustiť nasledovne:

./kry -g B

Výstup: **P Q N E D,** pričom:

- **B**: požadovaná veľkosť verejného modulu N v bitoch
- **P**: veľké prvočíslo (činiteľ čísla N)
- **Q**: veľké prvočíslo (činiteľ čísla N)
- **N**: N = P*X (verejný modulus)
- **E**: 1 < E < phi(n) tak, že gcd(e, phi(n)) = 1 (verejný exponent)
- **D**: inv(e, phi(n)) (súkromný exponent)

Implementácia generovania parametrov

Prvou časťou je vygenerovanie dostatočne veľkých prvočísiel P a Q. Toto generovanie prebieha pomocou funkcie `getRandomPrimeBites`, ktorá vygeneruje prvočíslo vo veľkosti B / 2 bitov, pričom dodržiava pripomienku uvedenú v zadaní – MSB musí mať hodnotu 1. Na overenie, či je vygenerované číslo prvočíslom, je použitý algoritmus Miller–Rabin. Samotné interné generovanie čísiel prebieha pomocou funkcie `mpz_urandomb`, pričom počiatočná seed hodnota pre generátor je získaná čítaním z `/dev/urandom`. Počas generovania prvočísiel je taktiež overované, či po ich vynásobení P * X, a teda vypočítaní verejného modulu N, je jeho dĺžka B. Ďalší krok predstavuje náhodné vygenerovanie čísla E, pričom musí platiť 1 < E < phi(n) tak, že gcd(e, phi(n)) = 1, pričom phi(n) = (P - 1) * (Q - 1).

Posledný krok predstavuje výpočet D, pričom jeho hodnota je výsledkom operácie nájdenia inverzného prvku (Multiplicative inverse). Implementácia algoritmu vychádza z publikácie J. Nechvatal - Public-Key Cryptography (NIST SP 800-2). Tento výpočet predstavuje posledný krok generovania parametrov, po ktorom sú jednotlivé parametre vypísané na štandardný výstup stdout.

Implementácia šifrovania správy

Šifrovanie správy prebieha pomocou verejného kľúču a funkcie `mpz_powm` nasledovne:

$C = M^E \bmod N$

Implementácia dešifrovania správy

Dešifrovanie správy prebieha pomocou súkromného kľúča a funkcie `mpz_powm` nasledovne:

$M = C^D \bmod N$

Implementácia prelomenia algoritmu RSA

Prelomenie algoritmu RSA prebieha v dvoch krokoch. Prvý krok predstavuje metóda „triviálneho“ (pokusného) delenia číslami do 1000000. V prípade, že nájdeme číslo, ktoré delí vstupný verejný modulus N, overíme, či sú jeho oba prípadné delitele prvočísla. Tento fakt je potrebné overiť z dôvodu, že číslo N je vypočítané ako N = P * Q. V prípade, že sme úspešne našli jeho prvočíselných deliteľov, nepokračujeme sofistikovanejšou metódou. Avšak, ak táto triviálna metóda nebola úspešná, postupujeme pomocou Fermatovej Faktorizačnej metódy. Táto metóda je implementovaná vo funkcii `fermatsFactorization` a je prevzatá zo zdroja uvedeného v zdrojovom kóde. Túto metódu som si vybral z dôvodu, že sa jedná o jednu z odporúčaných metód uvedených v projektovom zadaní. Metóda je taktiež priamočiara, no po testovaní sa ukázal fakt, že aj napriek tomu, že je uvedená ako jedna z odporúčaných metód v zadaní, nesplňuje požiadavky kladené na rýchlosť.