Kryptografie
Projekt č. 2

Nikola Valešová
xvales02@stud.fit.vutbr.cz

23. dubna 2018

1 Úvod

Cílem projektu je detailně prozkoumat asymetrickou šifru RSA a poté provést implementaci generování soukromého a veřejného klíče, šifrování, dešifrování a prolomení šifry pomocí faktorizačních metod aplikovaných na veřejný modul.

Jako implementační jazyk je použito C++ doplněné o funkce poskytované knihovnou GMP (GNU Multiprecision Library) pro umožnění výpočtů ve víceslovní aritmetice.

2 Generování klíčů

Prvním úkolem bylo generování soukromého a veřejného klíče asymetrického šifrovacího algoritmu RSA. Vstupem je požadovaná velikost veřejného modulu v bitech, B. Generování klíčů je implementováno tak, že jsou nejprve vygenerována dvě náhodná prvočísla p a q, první z nich na $\frac{B}{2}$ bitech, druhé na $B - bitCount(p)$. Po každém vygenerování náhodné hodnoty je otestováno, zda se jedná o prvočíslo. Pro detekci prvočísel byl implementován algoritmus Miller–Rabin.

V dalším kroku je vypočten veřejný modul n jako násobek p a q. Pro zajištění, aby na pozici *MSB* byla hodnota 1, je použit bitový *and* vypočteného čísla s číslem $1 << (\frac{B}{2} - 1)$ a výsledek je testován na rozdílnost od 0. Pokud vygenerované číslo nesplňuje tuto podmínku, generují se prvočísla p a q znovu a celý proces se opakuje. Pro vygenerování náhodných hodnot p a q je použita funkce `mpz_urandomb` a náhodné generování je na počátku inicializováno pomocí časového razítka pro získání různých hodnot při opakovaném spouštění.

Následně je vypočtena hodnota $\phi = (p-1)\cdot(q-1)$, která je použita pro vygenerování veřejného exponentu e. Ten je náhodně vygenerován z rozsahu $\langle 1, \phi \rangle$ a musí pro něj platit, že je s hodnotou ϕ nesoudělný, tedy že $gcd(e, \phi) = 1$. Pro výpočet největšího společného dělitele dvou čísel slouží v knihovně GMP funkce `mpz_gcd(mpz_t rop, const mpz_t op1, const mpz_t op2)`. Ta byla ovšem ze zadání zakázaná, a proto je implementace Euklidova algoritmu na výpočet největšího společného dělitele součástí projektu, konkrétně se jedná o funkci `GCD(mpz_t a, mpz_t b, mpz_t c)`.

Poslední počítanou hodnotou je soukromý exponent d, který je určen jako multiplikativní doplněk e v modulo ϕ.

Výstupem jsou hodnoty p, q, n, e a d na jednom řádku oddělené mezerami.

2.1 Algoritmus Miller–Rabin

Miller–Rabinův test prvočíselnosti je metoda zkoumání a určování, zda je dané číslo prvočíslem. Je založena na pravděpodobnostním usuzování a má dva vstupní parametry, testované číslo a počet iterací určující přesnost. Výstupem je buď rozhodnutí, že dané číslo je číslo složené, nebo domněnka, že je pravděpodobně prvočíslem. Čím více takových iterací provedeme, tím nižší je chyba daného algoritmu. V aplikaci je počet iterací nastaven na hodnotu 10, která je ve mnoha zdrojích považována za optimální a poskytuje přesnost přibližně 4^{-10}, což je v rámci tohoto projektu dostačující.

3 Šifrování a dešifrování

Druhým a třetím úkolem je provést šifrování při zadaném veřejném klíči a hodnotě nezašifrované zprávy a dešifrování, ve kterém je vstupem soukromý klíč a zašifrovaná zpráva. Samotný algoritmus pak v obou případech pracuje stejně, a to $message^{exponent}\ mod\ modulus$. Výsledná hodnota je na závěr vypsána na výstup.

4 Prolomení RSA

Posledním úkolem bylo prolomení slabého klíče RSA faktorizací veřejného modulu. Samotná faktorizace sestává ze dvou částí – nejprve je proveden pokus o faktorizaci dělením prvními 1 000 000 čísli. Takto lze získat jeden faktor modulu, pokud bude klíč dostatečně slabý.

Tento algoritmus lze značně zrychlit, pokud se budeme pokoušet dělit pouze prvočísly. Jelikož dělíme postupně všemi prvočísly od 2, dělení složeným číslem by nám nikdy nepřineslo úspěšné prolomení. Původně byla pro získání následujícího prvočísla implementována funkce, která aktuální prvočíslo inkrementovala a testovala, zda se jedná o prvočíslo, či ne. Tento přístup byl však relativně pomalý, proto byl výpočet dalšího prvočísla nahrazen polem staticky uložených prvočísel v souboru *kry.h*, nyní se tedy inkrementuje pouze index do tohoto pole. Pokud je nalezeno prvočíslo, které daný modul dělí, vypíše se na výstup a program končí.

Pokud zkusmé dělení neuspěje, je na modulus aplikována faktorizační metoda Pollard Rho. Zde byla původně implementována metoda Fermatova, avšak ta na větších číslech značně překračovala stanovený časový limit a metoda Pollard Rho napomohla značnému zrychlení.

4.1 Faktorizační metoda Pollard Rho

Jako pokročilejší metoda faktorizace byla zvolena metoda Pollard Rho, a to především díky její nízké prostorové složitosti a oproti metodě Fermatově i vyšší rychlosti. Základní myšlenkou této metody je iterativní výpočet určité rovnice, který prohíhá, dokud se nedostane do cyklu. Na vstupu metoda očekává hodnotu n, kterou chceme faktorizovat, a polynomiální funkci $g(x)$, která je počítána v modulo n. V implementaci je využita funkce $g(x) = (x^2+1) mod n$. Výstupem metody je potom buď neúspěch při faktorizaci, nebo hodnota jednoho netriviálního faktoru čísla n.

Algoritmus pracuje následujícím způsobem:

1. inicializace proměnných – proměnné x a y jsou inicializovány na hodnotu 2, proměnná d na číslo 1
2. iterační výpočet funkce $g(x)$ – dokud je d různé od 1, vypočti $g(x)$ a ulož výsledek do proměnné x, vypočti $g(g(y))$ a výsledek ulož do proměnné y a do proměnné d ulož hodnotu největšího společného dělitele čísel n a $|x - y|$
3. po opuštění předchozího cyklu jsou porovnány hodnoty d a n, pokud jsou si rovny, znamená to neúspěšnou faktorizaci, pokud jsou rozdílné, představuje hodnota v proměnné d právě vypočtený faktor

5 Závěr

V rámci tohoto projektu byl nastudován asymetrický šifrovací algoritmus RSA a byla naimplementována aplikace schopná provádět generování klíčů na daném počtu bitů, šifrovat a dešifrovat zprávy a prolomit slabou šifru rozložením veřejného modulu na dva faktory.