VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Dokumentace 2. projektu pro předmět KRY 2015/2016

Daniel Žůrek – xzurek12

1. Úvod

Cílem projektu je seznámit se s asymetrickým algoritmem RSA pro šifrování pomocí veřejného klíče a dále implementace tohoto algoritmu. Program bude schopen generovat parametry RSA (soukromý a veřejný klíč), šifrovat a dešifrovat. Dále bude umět prolomit RSA pomocí faktorizace slabého veřejného modulu. Vytvořený faktorizační nástroj bude schopen faktorizovat nejen slabé RSA klíče, ale jakékoli složené číslo do 96 bitů.

2. Popis algoritmu

i. Generuj dvě velká prvočísla p a q
ii. $n = p * q$
iii. $\Phi(n) = (p - 1) * (q - 1)$
iv. Zvol náhodně e mezi 1 a $\Phi(n)$; $gcd(e, \Phi(n)) = 1$
v. Vypočítej $d = inv(e, \Phi(n))$; $inv = e^{-1} \, mod \, \Phi(n)$
vi. Veřejný resp. soukromý klíč je potom dvojice (e, n) resp. (d, n)

3. Generování klíčů

Pro vygenerování klíčů p, q, n, $\Phi(n)$, e a d slouží funkce `generateKeys()`. Nejdříve je nutné vygenerovat dvě prvočísla p a q na základě počtu bitů výsledného modulu n. Prvočísla jsou generována náhodně pomocí pseudonáhodného generátoru Mersenne Twister, přítomného v knihovně GMP. Jelikož nelze spoléhat, že generátor vygeneruje prvočíslo, je nutné provést test prvočíselnosti. Není-li vygenerované číslo prvočíslem, je inkrementováno o 1 a test se opakuje pro každé další liché číslo. Tímto je docíleno efektivnějšího vyhledání jednoho z dalších prvočísel nacházejících se v intervalu od n až $2n - 2$ (Bertrandův postulát).

Pro určení je-li dané číslo prvočíslo, existuje několik metod např. metoda Solovay-Strassen, nebo metoda Miller-Rabin. V projektu je použita metoda Miller-Rabin ve funkci `rabinMiller()`. Jedná se o algoritmus pracující v polynomiálním čase, kde přesnost výsledku je dána počtem iterací.

Následně je vypočten modul n, který je odvozen z kroku ii. v kapitole 2. Dle kroku iii. a iv. je zvolena hodnota $\Phi(n)$ a e. Funkce pro nejmenší společný dělitel dvou čísel je implementována ve funkci `myGcd()`, jelikož zadání některé knihovní funkce zakazuje. Klíč d je vygenerován na základě kroku v., pomocí funkce `mpz_powm()`, která umocní bázi (e) na daný exponent (-1) a nad výsledkem provede operaci modulo v dané zbytkové třídě ($\Phi(n)$).

4. Šifrování a dešifrování

Šifrování pomocí veřejného klíče:

$$c = m^e \bmod n$$

Dešifrování pomocí soukromého klíče:

$$m = c^d \bmod n$$

c – zašifrovaný text (celé číslo)
m – původní zpráva

Šifrování/dešifrování je implementováno ve funkcích `encrypt()` a `decrypt()`. V obou funkcích je použita metoda `mpz_powm()`.

5. Prolomení RSA

Prolomení RSA šifry je založeno na problému faktorizace, tedy rozložení čísla na součin menších čísel. V případě RSA rozklad celého čísla na součin prvočísel. Předpokládá se, že faktorizace není jednoduchý problém, na čemž je bezpečnost šifry založena.

Implementace faktorizační části

Faktorizační část sestává ze dvou částí. Veřejný modulus se nejdříve zkontrolujte pomocí metody triviálního dělení pro prvních 1 000 000 čísel. Triviální dělení je nejjednodušší metoda pro faktorizaci celých čísel. Metoda pracuje na základě zvolení počátečního dělitele a následném ověření, zda dělitel opravdu dělí zadané číslo beze zbytku. Pokud ne, dojde k inkrementaci hodnoty dělitele a opět se ověřuje dělitelnost beze zbytku. Tento proces je opakován tak dlouho, dokud není nalezen správný dělitel. Jelikož se jedná o metodu s exponenciální časovou složitostí, je velice pomalá při faktorizaci větších čísel, proto se používá jen na rychlé ověření dělitelnosti nějakým malým faktorem. Triviální dělení je implementováno ve funkci `trialDivision()`.

Pokud se pro dané číslo nenalezne výsledek pomocí triviálního dělení, je třeba použít sofistikovanější metodu. Jako první jsem zkoušel implementovat Fermatovu faktorizační metodu, zde jsem však narazil na problém při operaci odmocniny, kdy docházelo k velké chybě díky malé přesnosti.

Další možnou metodou pro faktorizaci je např. metoda Pollard rho. Máme-li kladné celé číslo n, o kterém víme, že je složené. Musí tedy existovat alespoň dvě čísla, jejichž vynásobením dostaneme n. Jedno z těchto čísel, označme jej d, musí být menší nebo rovno $\sqrt{n}$. Budeme-li náhodně vybírat čísla a, b z této množiny, jediná možnost jako získat $a \equiv b \ (mod\ n)$ je, pokud $a = b$. Jelikož je ale d menší než $\sqrt{n}$, je vysoká pravděpodobnost, že se nám podaří najít $a \equiv b \ (mod\ d)$ takové, aby $a \neq b$. Potom bude platit, že $|a - b|$ je násobek d, a jelikož n je také násobkem d, získáme faktor za pomoci největšího společného dělitele:

$$x = gcd(|a - b|, n)$$

Jako faktorizační metoda byla zvolena metoda Pollard rho, implementovaná ve funkci `pollardRho()`. Metoda byla zvolena díky své jednodušší implementaci, avšak při špatných

podmínkách je metoda Pollard rho velmi pomalá. Možností vylepšení je použití Brentovi metody.

6. Závěr

Byl implementován asymetrický šifrovací algoritmus RSA. Program je schopen generovat RSA klíče potřebné pro šifrování a dešifrování dané zprávy. Dále je program schopen prolomit šifru na základě slabého veřejného modulu. Jako faktorizační metoda byla kromě metody triviálního dělení implementována metoda Pollard rho. Pro prvočíselné testování byla implementována metoda Miller-Rabin. Program je napsán v jazyce C za použití knihovny GMP. Projekt byl testován na školním serveru Merlin.