Projekt z predmetu KRY Martin Maga (xmagam00)
Implementace a prolomení RSA

1 Teoretické rozobratie algoritmu

AAlgoritmus RSA patrí medzi asymetrický algoritmus, čo znamená, že pre šifrovanie a dešifrovanie správy sa používajú 2 typy kľúčov: Privátny kľúč a verejný kľúč. Pri zašifrovanie správy sa používa verejný kľúč a pre dešifrovanie sa používa klúč privatný. V prípade použitia tohto algoritmu sa používa vžd kombinácia (e, n), kde e je verejný exponnet a n verený modulus, pričom táto kombinácia sa nazýva verejný kľúč. Dvojica (d, n), kde d je privátny exponent a n je verejný modulus, pričom táto kombinácia sa nazýva privátny kľúč. Postup získania týchto kľúčov bude ukázané v časti implementácia. Tento algoritmus je považovaný za bezpečný pokiaľ sa splnia všetky podmienky pri generovaní verejného a súkromného klúča.

2 Implementácia

Pri implementácií projektu som používal jazyk C++ spolu s knižnicou GMP, ktorú som použil pre prácu s veľkými číslami. Treba podotknúť, že všetky čísla na vstupe a výstupe boli v hexadecimálnom formáte okrem časti pre generovanie klúčov, kde sme zadávali požadovanú dĺžku klúčov v desiatkovej sústave.

2.1 Šifrovanie

Šifrovanie bolo realizované prostredníctvom funkcie "$gmp_p own$", ktorá spracovala vstupné parametre(otvorená správa, verejný modulus a verejný exponent) a tieto podľa vzťahu vypočítala: c[1] $= m^{e 2}$ mod n[3]. Táto funkcia nám priamo umocní a urobí modulu a výsledok uloží do jej 1. parametre. Výsledok sa vypíše na štandardný výstup.

2.2 Dešifrovanie

Dešifrovanie bolo realizované úplne rovnakou funkciou ako šifrovanie ale vstupe bol súkromný exponent, zašifrovaná správa a verejný modulus. Pri použití funkcie "$gmp_p own$" sme vypočítali otvorenú správu podľa vzťahu. m[4] $= c^{d 5}$ mod n[6].

2.3 Generovanie klúčov

Pri generovaní klúčov bola na vstupe zadaná požadovaná dĺžka verejného modulu v bitoch, napr. 1024, 512 bitov atď. Vieme, že verejný modulus dostaneme súčinom 2 rozličných prvočísel.
Celý postup sa realizuje nasledovne. Najprv vygenerujeme p a q nasledovne: (B+1)/2-bitové p a (B - (B+1)/2)-bitové. Nastavím najvyššie bity na jednotku, čím zabezpečím, že budú mať požadovanú dĺžku. Vieme, že 2 512 bitové čísla môžu dať ich súčinom vo výsledku až 1024 bitové číslo. Teda v prípade, že požadovaná dĺžka verejného modulu je 1024 bitov vytvorím p

[1] Predstavuje zašifronvaú správu
[2] M je otvorená správa a E je verejný exponent
[3] verejné modulo
[4] Otvorená správa
[5] šifrovaná správa umocnená na súkromný exponent
[6] verejný exponent

a q podľa vzťahu uvedeného vyššie a nastavím najvyššie vity na hodnotu 1. Následne potrebujeme zabezpečiť, aby čísla p a q boli odlišné a zároveň boli prvočísla. Pre test, či sú obe čísla prvočísla používam algoritmus Solovan Strayssen. Týmto algoritmom overíme, či vygenerované čísla p a q sú prvočísla. V prípade, že čísla nie sú prvočísla náhodne vygenerujeme nové čísla, nastavíme najvyšší bit na hodnotu 1 a overíme opäť či sú čísla prvočísla.
V prípade, že sú čísla prvočísla vypočítame hodnotu verejného modulu n podľa vzťahu: n = p * q. Ďalej vypočítam hodnotu phi(n) = (p - 1) * (q - 1). Následne vygenerujem súkromný exponent "e". Tento má hodnotu buď 3 alebo $2^16 + 1$. Buď sa zvolí jedna alebo druhá hodnota, pričom musí platiť podmienka: gcd(e, phi(n)) = 1. Funkcia gcd predstavuje najväčší spoločný deliteľ, pričom treba zabezpečiť, aby hodnota phi(n) a verejný exponent a boli nesúdelitelné. Funkcia gcd je implementovaná euklidovým algoritmom, ktorý nájde najväčšieho spoločného deliteľa 2 čísel. Nakoniec je nutne určiť hodnotu súkromného exponentu d. d = inv(e, phi(n)) - inv je operácia nájdená inverzného prvku tzv Multiplicate inverse. d = e^-1 mod n.
Po tomto kroku máme vypočítané všetky potrebné parametre pre algoritmu a preto ich zobrazíme na výstup.

2.4 Faktorizácia - neimplementované

Program by mal byť schopný faktorizovať slabý verejný modul do 96 bitov. Rovnako by mal byť schopný faktorizovať slabé RSA klúče.
V princípe najprv použijeme najprv triviálnu metódu, kde začíname s počiatočnou hodntou 2 a postupne delíme číslo, ak dostaneme číslo bez zbytku máme rozklad číslo v opačnom prípade, číslo inkrementujeme a postup opakujeme až do čísla 10.
Ďalej by som použil metódu "Fermat's Factorization Method". Táto metóda je založená na na reprezentácií nepárneho čísla ako rozdiel 2 štvorcových hodnôt. N = $a^2 - b^2$, čo môžme rozpísať ako (a+b)*(a-b). Táto metóda môže dosahovať horšiu časovú zložitosť ako naivné riešenie popísané vyššie. Preto sa používa kombinácia triviálneho riešenia spolu s touto metódou.
FermatFactor(N): // N should be odd
a = ceil(sqrt(N))
b2 =a*a - N
while b2 is not a square:
a = a + 1 // equivalently: b2 = b2 + 2*a + 1
b2 = a*a - N // a = a + 1
endwhile
return a - sqrt(b2) // or a + sqrt(b2)

3 Záver

Implementovali sme šifrovanie a dešifrovanie správ prostredníctvom algoritmu RSA. Rovnako sme implementovali generovanie privátneho a verejného klúča. Rovnako som načrtoval spôsob faktorizácie verejného modulu, ktoré viedlo k prelomeneniu algoritmu RSA.