Dokumentácia do predmetu Kryptografia: Implementácia a prelomenie RSA

Meno: Juraj Ondrej Dúbrava
Login: xdubra03

Implementácia RSA

Algoritmus RSA je asymetrický, základným princípom je pozorovanie, že je praktické nájsť tri veľké čísla e,d,n také, že platí $(m^e)^d \equiv m \pmod n$, teda tieto čísla sú kongruentné modulo n, pre všetky 0<= m < n. Algoritmus RSA pozostáva z niekoľkých krokov: získania dvoch veľkých prvočísel p a q, získania hodnoty verejného modulu *n*, získania hodnoty φ a výpočtu verejného a súkromného exponentu, ktoré sú použité pri šifrovaní a dešifrovaní správy. Pri implementácii bola použitá knižnica GMP. Na začiatku sa vygenerujú dve náhodné čísla *p* a *q* a následne je potrebné overiť, či tieto čísla sú prvočísla. Seed použitý v generátore náhodných čísel beriem ako hodnotu zo systémového času, čo však nie je najideálnejšie. Na overenie či je dané číslo prvočíslo existuje niekoľko metód, z ktorých som si zvolil metódu Solovay-Strassen[1]. Ide o pravdepodobnostnú metódu, kde sa overuje či platí kongruencia $a^{(p-1)/2} \equiv (a/n) \bmod n$. Opakovane testujeme platnosť pre rôzne hodnoty *a; a<n* , ak zistíme pre nejaké a že vzťah neplatí, *n* nie je prvočíslo. Na zaručenie dostatočnej dôveryhodnosti výsledku sa overuje v implementácií 50 rôznych hodnôt a. Ako ďalší krok vypočítame hodnotu verejného modulu ako *n=p*q*. Ďalej vypočítame hodnotu φ = (p-1) * (q-1), a ideme vygenerovať hodnotu verejného exponentu *e* v rozmedzí 1 a φ(n), tak aby platilo gcd(e,φ(n))=1. Opakovane teda generujem hodnotu e a pomocou Euklidovho algoritmu [2] sa počíta hodnota gcd. Po získaní hodnoty verejného exponentu zostáva vypočítať hodnotu súkromného exponentu *d*. Ten sa získa nájdením inverzného prvku, kde sa počíta multiplikatívny inverz hodnoty e modulo φ(n). Na výpočet sa používa rozšírený Euklidov algoritmus [3]. Ten okrem hodnoty gcd zadaných čísel počíta aj koeficienty tzv. Bézoutovej identity také, že platí

$$ax + by = \gcd(a, b).$$

Ak a a b sú voči sebe relatívne prvočísla, teda gcd(a,b) = 1, v tom prípade je koeficient x modulárnym multiplikatívnym inverzom a modulo b. Dokončením tohto kroku máme vygenerované všetky parametre RSA, dvojica (*e,n*) je verejný kľúč a dvojica (*d,n*) je súkromný kľúč. Šifrovanie prebieha ako operácia $m' = m^e \pmod n$, dešifrovanie ako $m' = m^d \pmod n$.

Prelomenie RSA

Prelomenie RSA spočíva v zistení prvočísel *p* a *q*, ktoré musíme zistiť z hodnoty verejného modulu *n*. Modulus je však číslo tvorené násobením prvočísel *p* a q, a ak chceme zistiť obsah pôvodnej správy, musíme nejakým spôsobom hodnoty *p* a *q* získať. Ide však o proces rozkladu čísla na menšie celky, ktorý je veľmi náročný. Tento proces sa nazýva faktorizácia a existuje niekoľko techník ako ju uskutočniť.

Faktorizácia verejného modulu

Jednou z faktorizačných techník je Pollardov ρ algoritmus [4] na faktorizáciu celých čísel, ktorý som si vybral ako spôsob faktorizácie verejného modulu. Výhodou je, že vyžaduje malé množstvo pamäti a potrebný čas je približný odmocnine menšieho z faktorov.
Tento algoritmus využíva niekoľkých princípov na zistenie faktoru, hlavnými sú detekcia cyklu Floydovým algoritmom a generovaním pseudonáhodných čísel určitým spôsobom.
Modul *n* chceme rozložiť na čísla *n* = *pq*. Pri výpočte sa použije polynóm g(x) modulo n, určený na generovanie pseudonáhodnej sekvencie čísel. Odporúčaným polynómom je g(x) = (x^2 + c) mod *n*, kde *c* je pseudonáhodne zvolená konštanta. Sekvencia produkovaná týmto polynómom bude spojená s inou sekvenciou {x_i mod p}. Keďže p dopredu nepoznáme, nevieme určiť čísla v tejto sekvencii. Obidve sekvencie, {x_k} aj {x_k mod p} sú konečné a po určitom počte prvkov sa budú opakovať. Za predpokladu, že sa sekvencie chovajú ako náhodné a vďaka tzv. narodeninovému paradoxu, číslo x_k by pred začatím opakovania malo dosiahnuť hodnotu O(√N), N je počet možných hodnôt. Sekvencia {x_k mod p} sa začne opakovať ale podstatne skôr ako {x_k}, tým že sa sekvencie začnú opakovať, vytvárajú cyklus so štruktúrou podobnou práve na písmeno ρ. Tu sa využíva ďalší z princípov, konkrétne Floydov algoritmus na detekciu cyklu. Ten je založený na tom, že ak zajac a korytnačka vyjdu naraz z rovnakého bodu a pohybujú sa v cykle tak, že rýchlosť zajaca je dvojnásobná ako rýchlosť korytnačky, musia sa po istom čase stretnúť v rovnakom bode. Po každom kroku sa kontroluje či gcd(x_i - x_j , n) nie je 1. Ak táto rovnosť nie je splnená, implikuje to, že v sekvencii {x_k mod p} je opakovanie. Toto funguje kvôli tomu, že ak napríklad x_i mod p je rovnaké ako x_j mod p, rozdiel x_i - x_j je určite násobok čísla p, ale aj *n* je násobkom p. Platí tam teda kongruencia x ≡ y (mod p). Najväčší spoločný deliteľ je teda výsledkom algoritmu, čo nám dá jeden z faktorov. Algoritmus môže zlyhať pri hľadaní faktoru, ak sa dostaneme až ku gcd takému, že sa rovná číslu n, je potrebné opakovať výpočet s inými parametrami.

Dešifrovanie

Po získaní jedného z faktorov stačí druhý získať delením hodnoty verejného modulu získaným faktorom. Následne vypočítame hodnotu φ = (p - 1) * (q - 1). Ďalším krokom je nájdenie hodnoty súkromného exponentu *d* tak, že musíme nájsť multiplikatívny inverz verejného exponentu *e* modulo φ, čiže inv(e, φ(n)). Tak sa získa *d* také, že platí e × d ≡ 1 (mod φ(*n*)).
Posledným krokom je uskutočnenie operácie $m'' = m'^d$ (mod *n*), čo je aj $m'' = m^{e * d}$ (mod *n*), zašifrovaná správa vznikla ako $m' = m^e$ (mod *n*). Pôvodná správa je $m'' = m$.

Zdroje:
[1] Solovay-Strassen test:
https://en.wikipedia.org/wiki/Solovay%E2%80%93Strassen_primality_test#Algorithm_and_running_time
[2] Euklidov algoritmus: https://en.wikipedia.org/wiki/Euclidean_algorithm
[3] Rozšírený Euklidov algoritmus: https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm
[4] Pollardov Rho algoritmus: https://en.wikipedia.org/wiki/Pollard%27s_rho_algorithm