Kryptografie – dokumentace k projektu 2

Vysoké učení technické v Brně

Petr Stehlík <xstehl14@stud.fit.vutbr.cz> 6. května 2018

1 Zadání

Cílem projektu bylo vytvořit program pro generování parametrů pro algoritmus RSA, šifrování a dešifrování zprávy. Program také implemetuje prolomení RSA pomocí faktorizace slabého veřejného modulu a jakéhokoliv složeného čísla do 96 bitů.

2 Implementace generování parametrů

Algoritmus generování parametrů P, Q, N, E a D pro RSA pracuje následovně:

- Generuj dvě velká prvočísla P a Q
- $N = P * Q$
- $\phi(n) = (P - 1) * (Q - 1)$
- Zvol náhodně E mezi 1 a $\phi(N)$ tak, že $gcd(e, \phi(N)) = 1$
- Vypočítej $D = inv(E, \phi(N))$ - inv je operace nalezení inverzního prvku
- Veřejný klíč je dvojice (E, N)
- Soukromý klíč je dvojice (D, N)

Pro generování korektních parametrů P, Q a N platí, že pokud požadujeme N o velikosti n bitů, musí být parametry P a Q z intervalu $(\sqrt{2} \times 2^{n/2-1}, 2^{n/2})$[1].

Pro generování čísel v daném rozsahu byla použita metoda třídy `gmp_randclass.get_z_range` z knihovny GMP následujícím způsobem: `get_z_range(end - start) + start`. Kde `start` a `end` označují začátek, resp. konec intervalu.

Tyto čísla následně byla zkontrolována metodou Miller-Rabin zda jsou prvočísla. Pokud nebyly, generovaly se nová čísla dokud nebylo generované číslo prvočíslem.

Následně se vypočítalo ϕ a E, kde pomocí euklidovy metody hledání největšího společného dělitele[2] se generoval vhodný parametr E.

Parametr D se získal operací nalezení inverzního prvku rozšířeným euklidovským algoritmem[3].

Tímto postupem byly získány všechny potřebné parametry pro šifrování a dešifrování pomocí algoritmu RSA, které jsou v hexadecimální podobě vypsání na standardní výstup oddělené mezerou v daném pořadí: P, Q, N, E, D.

3 Implementace šifrování a dešifrování

Pro šifrování algoritmem RSA je nutná dvojice (E, N) a zpráva M. Šifrování probíhá následovně:

$$C \equiv M^E \mod N.$$

Dešifrovací algoritmus je následující:

$$M \equiv C^D \mod N.$$

V obou algoritmech je C označením zašifrované a M nezašifrované zprávy. Pro odstranění side-channel útoků byla použita bezpečná funkce `mpz_powm_sec`[4].

[1] `https://bit.ly/2KHkVVi`

[2] `https://en.wikipedia.org/wiki/Euclidean_algorithm`

[3] `https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm#Modular_integers`

[4] `https://gmplib.org/manual/Integer-Exponentiation.html`

4 Implementace faktorizace

Faktorizace je tvořena dvěma algoritmy. První, naivní, algoritmus vyzkouší všechna lichá čísla do 1000000 zda nejsou celočíselným dělitelem daného čísla.

Pro komplexnější řešení byl zvolen algoritmus Pollard Rho[5] kvůli své efektivitě a jednoduchosti implementace. Tento algoritmus již využívá teorie čísel a ze vztahu $N = P \times Q$ hledá faktor P. Dále disponuje funkcí $g(x) = (x^2 + 1) \mod N$, která generuje pseudo-náhodnou posloupnost čísel. Tato posloupnost je konečná a tudíž je nutné detekovat zacyklení. To je kontrolováno pomocí Floydova algoritmu, který nalezne největšího společného dělitele absolutního rozdílu dvou po sobě následujících pseudonáhodných X a čísla N. Jakmile je tento faktor větší než 1, našli jsme hledané číslo P.

Hledaný faktor je následně vypsán jako hexadecimální číslo na standardní výstup a program je ukončen.

5 Závěr

Program úspěšně implemetuje generování parametrů pro algoritmus RSA, šifrování a dešifrování zprávy a faktorizaci parametru N. Program pro testované vstupy vždy faktorizoval 96bitová čísla do 120 sekund od spuštění programu.

[5] `https://en.wikipedia.org/wiki/Pollard's_rho_algorithm`