Kryptografie 2020
Projekt č. 2 – Implementace a prolomení RSA

Kočica Filip – xkocic01

1 Úvod

Projekt je implementován v jazyce `C++` za pomoci knihovny GNU MP[1] pro práci s velkými čísly.

Implementovány byly všechny možnosti, tj. generování, dešifrování, šifrování i prolomení RSA.

2 Generování klíčů

Začíná se generováním velkých prvočísel `P` a `Q`. Pro vygenerování čísla bylo použit generátor `mpz_urandomm()` a pro test, zda-li je vygenerované číslo prvočíslem byl implementován pravděpodobnostní algoritmus **Miller Rabin**[2]. Kontrola trvá poměrně krátkou dobu – časová složitost $O(k \cdot log_3(n))$, ale může se stát (po dobu testování se to nestalo), že algoritmus rozhodne o daném čísle, že je prvočíslo, přicemž ve skutečnosti prvočíslo není (je to pouze pravděpodobnostní test).

Poté se spočte `phi(n)` a provádí se hledání veřejného exponentu `E`. Pro tyto účely bylo implementováno hledání největšího společného dělitele (ang. *greatest common diviser*).

Dále se počítá soukromý exponent `D` za pomoci nalezení inverzního prvku (ang. *multiplicative inverse*) pomocí funkcí `Inverse()` a `Update()` implementovaných podle knihy *Public key cryptography*[2].

Nakonec se provede vypsání `P Q N E D` na výstup v hexadecimální soustavě.

Velikost vstupu je omezena na **alespoň 31 bitů** dlouhý veřejný modulus.

Velikost klíče [b]	Doba generování [s]
512	0.047
1024	0.280
2048	4.675
4096	177.523

3 Šifrování

Šifrování zprávy (převedené na číslo) `M` se provede následovně (pomocí `mpz_powm()`):

$$c = m^e \mod n,$$

čímž se získá šifrovaná zpráva a ta je vypsána na standardní výstup v hexadecimální soustavě.

4 Dešifrování

Dešifrování textu `C` za účelem získání původní zprávy `M` se provede následující (opět pomocí `mpz_powm()`):

$$m = c^d \mod n,$$

čímž se získá původní zpráva a ta je vypsána na standardní výstup v hexadecimální soustavě.

5 Prolomení RSA

Prolomení RSA na základě slabého veřejného modulu bylo implementováno pomocí Pollardovi faktorizační metody. Tato metoda je vhodná k rozložení **velkých** složených čísel na jejich prvočíselný rozklad, pokud je jedno z prvočísel značně menší. To v případě této implementace zcela neplatí, protože oba klíče `P` a `Q` pro generování veřejného modulu mají stejnou délku (polovinu délky veřejného modulu, zadanou při generování).

Algoritmus
Vstup: Testované složené číslo p.
Výstup: Prvočíselné faktory čísla p.

1. Vytvoř množinu $A = \{0, 1, ..., p-1\}$.

   Vyber libovolné $x_0 \in A$.

   Polož $y_0 = x_0, i = 0$.

2. Vypočti $x_i = f(x_{i-1})(mod p)$.

   Vypočti $y_i = f(y_{i-1})(mod p)$.

   Polož $g = nsd(|y_i - x_i|, p)$.

3. `if` $g \neq 1$ `return` "g je dělitel čísla p".

   `else if` $x_i = y_i$ `exit`

   `else` $i = i + 1$, `goto` 2.

Velikost klíče [b]	Doba faktorizace [s]
60	0.061
80	3.485
90	34.156
100	278.167

6 Závěr

Testování probíhalo převážně na referenčním stroji – školním serveru merlin. Generování klíčů je plně funkční i pro větší délky veřejného modulu (např. 1024, 2048, 4096) a je poměrně rychlé. Šifrování a dešifrování jsou funkční a rychlé operace (pokud je šifrovaná zpráva číslo). Prolomení RSA je poměrně pomalé z důvodu nevyužití největší výhody Pollardovi faktorizační metody, a to značně menší jeden z prvočíselných faktorů – po takovémto "správném" vygenerování klíčů by však implementace metody měla být poměrně rychlá.

7 Přílohy

Obrázek 1: Ukázka generování, šifrování, dešifrování i prolomení RSA při délce veřejného modulu 80 bitů.

```
xkocic01@merlin: ~/KRY$ ./kry -g 80
0xdf5c732865 0xe08a1d9201 0xc3e9666648cb0eedc265 0x4eb1a6b3c7f1f02b7321 0x2f22c03f4f0f4f4f58e1
xkocic01@merlin: ~/KRY$ ./kry -e 0x4eb1a6b3c7f1f02b7321 0xc3e9666648cb0eedc265 0x1234567890abcdef
0x826254c0c9a8ac27a0c7
xkocic01@merlin: ~/KRY$ ./kry -d 0x2f22c03f4f0f4f4f58e1 0xc3e9666648cb0eedc265 0x826254c0c9a8ac27a0c7
0x1234567890abcdef
xkocic01@merlin: ~/KRY$ ./kry -b 0x4eb1a6b3c7f1f02b7321 0xc3e9666648cb0eedc265 0x826254c0c9a8ac27a0c7
0xdf5c732865 0xe08a1d9201 0x1234567890abcdef
xkocic01@merlin: ~/KRY$
```

Odkazy

[1] GMP. *The GNU Multiple Precision Arithmetic Library*. URL: https://www.gmplib.org/.

[2] James Nechvatal. *Public Key Cryptography*. 1991. ISBN: SP 800-2. URL: https://csrc.nist.gov/publications/detail/sp/800-2/archive/1991-04-01.