KRY - Kryptografie 2020

Projekt č. 2 - Implementace a prolomení RSA

Zadání:

V tomto projektu se seznámíte s asymetrickým šifrováním, konkrétně s algoritmem RSA. Váš program bude umět generovat parametry RSA (soukromý a veřejný klíč), šifrovat a dešifrovat. Dále bude umět prolomit RSA pomocí faktorizace slabého veřejného modulu tak, že ze zašifrovaného bloku zprávy a veřejného klíče dokáže zjistit zprávu v otevřeném tvaru (bez předchozí znalosti soukromého klíče) pro takové parametry RSA, kde veřejný modulus n je dostatečně malý, aby to šlo provést v krátkém čase (velikost cca 96b, v praxi se používá min. 1024b). Cílem je ukázat důležitost používání dostatecně dlouhých klíčů.

Algoritmus RSA:

- 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 (Multiplicative inverse). Popis tohoto algoritmu najdete v textu J. Nechvatal - Public-Key Cryptography (NIST SP 800-2), který je dostupný z hlavní stránky kurzu.
- Veřejný resp. soukromý klíč je potom dvojice (e,n) resp. (d,n)

Generování resp. testování prvočísel pomocí metody Solovay-Strassen, případně Miller-Rabin je podrobně popsané v textu Public-Key Cryptography.

Požadavky

Program implementujte v jazyce C/C++. Pro implementaci RSA můžete použít knihovnu pro práci s velkými čísly (např. GMP), používejte ji však pouze pro základní operace s velkými čísly jako je například násobení nebo modulo,

nepoužívejte složitější operace, např. generování prvočísel (zejména nepoužívejte funkce typu mpz_nextprime, mpz_gcd nebo mpz_invert při použití GMP). Pro prolomení RSA můžete použít libovolnou knihovnu, která bude obsahovat potřebné algoritmy. V dokumentaci popište implementaci RSA jen velmi stručně, zaměřte se na prolomení RSA. Popište, jak váš program provádí faktorizaci a dešifrování, podrobně popište algoritmus, který používáte na faktorizaci veřejného modulu (i když bude součástí nějaké knihovny) a zdůvodněte, proč jste použili právě tento. V žádném případě nepoužívejte metodu naivního dělení, program musí být rychlý. Příliš dlouhý výpočet může vést k bodové srážce.

Testování a hodnocení

Generování klíčů (3b)
vstup: "./kry -g B"
výstup: "P Q N E D"

Šifrování (0.5b)
vstup: "./kry -e E N M"
výstup: "C"

Dešifrování (0.5b)
vstup: "./kry -d D N C"
výstup: "M"

Prolomení RSA (3b)
vstup: "./kry -b E N C"
výstup: "P Q M"

- B ... požadovaná velikost veřejného modulu v bitech (např. 1024)
- P ... prvočíslo (při generování náhodné)
- Q ... jiné prvočíslo (při generování náhodné)
- N ... veřejný modulus
- E ... veřejný exponent (většinou 3)
- D ... soukromý exponent
- M ... otevřená zpráva (číslo, nikoli text)
- C ... zašifrovaná zpráva (číslo, nikoli text)
- všechna čísla na vstupu i výstupu (kromě B) jsou hexadecimální, začínají "0x"
- výstup končí novým řádkem

- v případě, že danou funkci neimplemetujete, program skončí bez výstupu

Příklad:
vstup:

./kry -b 0x3 0x2f72f3e766a6ee4497ec836160e05363be3c0eb5adbdddb47
0x850e13a7c8826f9a8682f611dffe8cadd75e57a351ab9864

výstup:
0x382902d2f58d2754b39bca8f5 0xd84a873b14b9c0e1680bbdcb 0xaa3dc18b1eff7e89bf7678636580d1c3

Odevzdání

Do wisu odevzdávejte archiv xlogin00.zip se zdrojovými kódy vaší aplikace. Aplikace musí obsahovat makefile, který po zavolání příkazu "make" vytvoří spustitelný soubor "kry".

Dotazy k projektu směřujte primárně na fórum. Konzultace poskytuje: malinka@fit.vutbr.cz
Datum odevzdání: 3.5.2020