
DOKUMENTÁCIA [STUDENT] – Študijné materiály

Autor: Martin Janitor

Bakalárska práca: RSA s výplňovou schémou OAEP

Dátum: 05.06.2022

Verzia: 1.0

Štruktúra kryptografickej knižnice STUDENT:

EXTENSIONS_STUDENT

|---------- **include**

|---------- ext_bignum.h

|---------- ext_file.h

|---------- ext_rsa.h

|---------- **src**

|---------- ext_bignum.c

|---------- ext_file.c

|---------- ext_rsa.c

|---------- **TESTS**

|---------- Makefile

|---------- test_vect.c

|---------- test01.c

|---------- test02.c

|---------- test05.c

|---------- test06.c

STUDENT

|---------- **src**

|---------- bignum.c

|---------- bignum.h

|---------- changes.txt

|---------- source.txt

OPIS KRYPTOGRAFICKEJ KNIŽNICE STUDENT

Link na pôvodnú implementáciu: https://github.com/junwei-wang/biginteger

Kryptografická knižnica STUDENT obsahuje implementáciu šifrovacieho algoritmu RSA s výplňovou schémou OAEP. Testovanie prvočíselnosti je realizované pomocou Miller-Rabinovho testu prvočíselnosti s parametrom **t = 80**, ktorý určuje počet iterácii testu prvočíselnosti. Implementácia obsahuje aj možnosť využitia výplňovej schémy OAEP v spojení s RSA algoritmom. Implementácia výplňovej schémy OAEP sa nachádza v adresári **/OAEP**. Implementácia výplňovej schémy OAEP využíva hashovacie funkcie SHA-1, SHA-256 a SHA-512, ktoré sú implementované v adresári **/HASH**.

Knižnica realizuje generovanie kryptograficky bezpečných náhodných čísel:

- Operačný systém Windows: využitá je funkcia rand_s()
- Operačný systém Linux: náhodné čísla sú načítavané zo súboru /dev/urandom

Formát reprezentujúci BN (Veľké číslo) číslo:

```
typedef struct {
    int sign;
    int size;
    block* tab;
} bignum;
```

Premenná **size** uchováva veľkosť alokovaného poľa **tab.**

Premenná **sign** uchováva hodnotu znamienka:

- 1 Kladné číslo.
- -1 Záporné číslo.

Typ **block** je zvolený podľa aktuálne využívanej kompilácie:

- 8-bitová kompilácia: block = int64_t
- 32-bitová kompilácia: block = _int128_t

Typ **var**, ktorý je definovaný v súbore bignum.h a je určený pre uloženie dát, pre aktuálne zvolenú kompiláciu (8-bitová alebo 32-bitová). Ak využívame 8-bitovú kompiláciu správu je potrebné uložiť do štruktúry typu pole, ktoré bude typu **unsigned char**. Ak využívame 32-bitovú kompiláciu, správu je potrebné uložiť do štruktúry typu pole, ktoré bude typu **unsigned int**. Typ **var** nám automaticky pridelí vhodný typ (unsigned int alebo unigned char) pre aktuálnu kompiláciu.

OPIS SÚBOROV

---

ext_bignum.c, ext_bignum.h

- Implementácia rozširujúca bn.h, bn.c (pôvodná verzia).
- Implementácia funkcií na konverziu správy na BN číslo a opačne, výpis BN čísla v hexadecimálnom tvare.
- Implementácia funkcií pre generovanie kryptograficky bezpečných náhodných čísel .

ext_file.c, ext_file.h

- Zápis a načítavanie RSA kľúčov a správ využitím súborov.

ext_rsa.c, ext_rsa.h

- Pridanie funkcie pre generovanie RSA kľúčov s využitím generovania kryptograficky bezpečných náhodných čísel.
- Zadefinovanie štruktúry RSA kľúča.

- Šifrovanie a dešifrovanie správy s algoritmom RSA a možnosťou využitia výplňovej schémy OAEP.

bignum.c, bignum.h

- Obsahujú funkcie na generovanie RSA kľúčov.
- Šifrovanie a dešifrovanie s využitím RSA algoritmu.
- Implementácie matematických operácii s BN číslami.

changes.txt - Opis vykonaných zmien v bignum.c a bignum.h .

source.txt - Odkaz na pôvodnú implementáciu.

VYUŽITIE 32-BITOVÉHO a 8-BITOVÉHO ZÁKLADU ČÍSLA

Pôvodná implementácia STUDENT bola rozšírená o možnosť vybratia číselného základu pre kompiláciu. Na vyber sú dva základy, **8-bitový** a **32-bitový** číselný základ.

Uveďme ako príklad uloženie BN čísla do pamäte PC, ktoré ma veľkosť **1024** bitov v jazyku C s využitím 32-bitového základu čísla. Je potrebné alokovať pole o veľkosti 1024/ 32 = 32 prvkov. Maximálna hodnota jedného prvku uloženého v pamäti je 2^32 -1 = 4294967295.

Pre uloženie BN čísla o veľkosti **1024** bitov s využitím 8-bitového základu čísla je potrebné alokovať pole o veľkosti 1024/8 = 128 prvkov. Maximálna hodnota jedného prvku uloženého v pamäti je 2^8 -1 = 255.

Počítače dokážu vykonávať matematické operácie s využitím 32 a viac bitových registrov. Implementácia s využitím 32-bitového základu čísla je efektívnejšia, pretože je potrebných menej krokov pri výpočte ako pri 8-bitovej implementácii.

TESTY

---

test01 Testuje overenie správnosti výpočtu matematickej operácie modulárneho umocnenia m^e mod n.

test02 Generuje RSA kľúče [1024, 2048, 4096 bitov] a šifruje správy s využitím RSA a výplňovej schémy OAEP + meranie času šifrovania a dešifrovania.

test05 Generuje RSA kľúče [1024, 2048, 4096 bitov] + meranie času generovania kľúčov.

test06 Test realizuje overenie zápisu správy do súboru a následne načítanie správy zo súboru a overenie.

test07 Vygeneruje RSA kľúče, ktoré následne uloží do súborov.

test08 Načíta zo súboru správu a RSA kľúče a následne zašifruje správu a odšifruje správu a porovná s pôvodnou správou zo súboru. Šifrovanie a dešifrovanie je realizované RSA algoritmom s využitím výplňovej schémy OAEP.

test09 Test realizuje generovanie náhodnej správy a náhodných RSA kľúčov v cykle. Testuje korektnosť výpočtu RSA algoritmu s výplňovou schémou OAEP.

test_vect Otestovanie RSA + OAEP s testovacími vektormi dostupných na stránke [https://www.inf.pucrs.br/~calazans/graduate/TPVLSI_I/RSA-oaep_spec.pdf] .

MAKEFILE

Vopred preddefinované MAKRA (možnosť využitia pri testoch):

- **COMP_8** [Kompilácia testu bude realizovaná s využitím 8-bitového číselného základu]
- **COMP_32** [Kompilácia testu bude realizovaná s využitím 32-bitového číselného základu]
- **TEST_VECT** [Pridanie do projektu testovacie vektory, ktoré sú zadané „napevno“ a sú priradené pri kompilácií projektu, ak je vopred zadefinované makro TEST_VECT.

Testovacie vektory: https://www.inf.pucrs.br/~calazans/graduate/TPVLSI_I/RSA-oaep_spec.pdf]

- **PRINT_TEST_VECT** [Vypis elementov pre kontrolu: p,q, n, e ...]

Pri každej kompilácií testu musí byt zadefinované makro **COMP_8** alebo **COMP_32** pre výber danej kompilácie. Vyššie uvedené testy majú už priradené MAKRA pre vyber kompilácie, pričom ich modifikáciou v súbore **Makefile** je možné zmeniť typ číselného základu.

Kompilácia testu:

make test[číslo testu] príklad: **make** test02

FORMÁTY PRE NAČÍTANIE A ZÁPIS DO SÚBOROV

V rámci rozšírenej implementácie bol navrhnutý aj špecifický formát pre načítavanie a uloženie správ a RSA kľúčov do súboru.

FORMÁT (správa):

1. Hlavička: **----BEGIN MSG-----** .
2. Postupnosť čísel reprezentujúca BN číslo: MSB bajt správy .... LSB bajt správy.
3. Päta: **-----END MSG----- .**

FORMÁT (RSA kľúče):

1. Hlavička**: -----BEGIN PRIVATE KEY-----** (pre súkromný kľúč) alebo **-----BEGIN PUBLIC KEY-----** (pre verejný kľúč).
2. Textový reťazec **[e]** (verejný exponent) alebo **[d]** (súkromný exponent).
3. Postupnosť čísel reprezentujúca BN číslo: MSB bajt .... LSB bajt (reprezentácia exponentu).
4. Textový reťazec **[n]**.
5. Postupnosť čísel reprezentujúca BN číslo: MSB bajt .... LSB bajt (reprezentácia modulusu).
6. Päta: **-----END PUBLIC KEY-----** (verejný kľúč) alebo **-----END PRIVATE KEY-----** (súkromný kľúč).

Príklady uloženia správy a RSA kľúčov v súbore:

Správa:

-----BEGIN MSG-----

D4 36 E9 95

69 FD 32 A7

C8 A0 5B BC

90 D3 2C 49

-----END MSG-----

Súkromný kľúč [d,n]:

-----BEGIN PRIVATE KEY-----

[d]

0A AF 2B A1 38 A1 F5 FE FE D8 4F E2 EC 2A 26 C5

22 50 1B CF 06 21 F2 DA FB B2 E2 86 1E 6E 7C 6F

6C 99 58 35 9F 8A C7 C8 CD 35 3E D3 F4 13 13 CD

55 96 53 93 3C 12 89 CD 87 1A 0E B5 08 1E AC 84

8D D8 B8 F8 95 8E 28 C6 A7 7F 05 57 F0 B6 09 83

05 01 7E 3E 0E ED CD 21 60 B6 99 AD 7E 13 FB 6F

64 F3 BA FC 04 68 07 23 05 5E 05 37 D4 E3 D2 1B

B2 BD 76 A8 12 B9 B6 9C C2 39 77 7D F8 35 E8 B1

[n]

0C 59 4B 4F F6 43 33 4E 31 DD E8 6B A6 65 63 33

0D 9E 1A 47 ED C3 C9 64 CD AA 18 23 F1 16 03 69

A9 D5 BC 8D D0 F7 96 02 81 72 7B 11 8B 3C 2F 83

DE 5F 62 47 97 E5 D7 49 48 74 94 3E 07 03 BC 90

E1 6E F5 3A 4F 6C 4F 99 74 03 16 53 F6 26 2A ED

64 C1 D4 0E 20 94 B8 B4 CA E9 A6 91 C2 79 82 4B

73 B9 88 9C BB 8E 59 01 54 09 3F A3 52 39 4E E7

42 99 80 2A 26 E3 8F 82 D2 91 C9 AF 0B 4C 68 7B

-----END PRIVATE KEY-----

Verejný kľúč [e,n]:

-----BEGIN PUBLIC KEY-----

[e]

01 00 01

[n]

0C 59 4B 4F F6 43 33 4E 31 DD E8 6B A6 65 63 33

0D 9E 1A 47 ED C3 C9 64 CD AA 18 23 F1 16 03 69

A9 D5 BC 8D D0 F7 96 02 81 72 7B 11 8B 3C 2F 83

DE 5F 62 47 97 E5 D7 49 48 74 94 3E 07 03 BC 90

E1 6E F5 3A 4F 6C 4F 99 74 03 16 53 F6 26 2A ED

64 C1 D4 0E 20 94 B8 B4 CA E9 A6 91 C2 79 82 4B

73 B9 88 9C BB 8E 59 01 54 09 3F A3 52 39 4E E7

42 99 80 2A 26 E3 8F 82 D2 91 C9 AF 0B 4C 68 7B

-----END PUBLIC KEY-----