Shchavleva Marina

Kontakt

shchamar@fit.cvut.cz (mailto:shchamar@fit.cvut.cz) (v předmětu použijte prefix **[BI-BEZ]**), nebo DM v Teams

Konzultace

Po předchozí domluvě

Domací úkoly

Hodnocení úkolů probíhá iterativně, máte „neomezený“ počet odevzdání, ale řešení odevzdaná po deadline již nezmění již získaný počet bodů (i nulový či záporný).

Každá iterace se skládá ze 3 částí:

1. Student/ka odevzdá úkol.
2. Cvičící ohodnotí a okomentuje odevzdaný kód.
    - Cvičící vrátí úkol k dopracování/přepracování.
    - Cvičící zmerguje úkol do `master` větve a máte hotovo.
3. Užívat si získané body.

Postup, jak odevzdávat domácí úkoly najdete na vlastní stránce (../krizjar5/homeworks.html) .

> ⚠ Každou větev prosím pojmenujte podle názvu složky na gitlab. Tzn. pro první úkol vytvořte větev (`git checkout -b nazev_ulohy`) s názvem úlohy `task1_mathematica`.

Deadlines

Hodina a minuta pro deadline je vždycky konec vašeho cvičení. Například pro paralelku pátek 9:15 deadline je v uvedený datum v 10:45.

Zadání	út 18:00	čt 7:30	pá 7:30, 9:15, 11:00
asymetrická kryptografie	27.4.	29.4.	30.4
síťová komunkace	11.5	13.5	14.5

Zadání

> ⚠ 💀 značí feature, bez kterých řešení nepovažuji za řešení a hodnotím nulou bodů. Pak [-Nb] značí feature, které nemusíte nutně mít, ale pokud bude chybět tak bude vám chybět N bodů v hodnocení.

Úkol 5: síťová komunikace

- jméno pro binárku není podstatné, ale bude brát 2 poziční parametry:

```
./connect FILECERT FILEPAGE
```

```
FILECERT   = arg       jméno souboru, kam se uloží certifikát
FILEPAGE   = arg       jméno souboru, kam se uloží stránka
```

- 💀 váš prográm nesegfaultuje
- 💀 návratové hodnoty: nula pro úspěch, 1 až 127 pro neúspěch.

Úkol 4: použití asymetrické šifry

- binárka se bude jmenovat `encrypt` a brát 4 poziční parametry v následující formě:

```
./encrypt ACTION PKEY FILEIN FILEOUT
```

```
ACTION      = {e,d}      má-li program provést šifrování (encrypt) nebo dešifrování (decr
PKEY        = arg        cesta ke {veřejnému, soukromému} klíči
FILEIN      = arg        cesta ke vstupnimu souboru
FILEOUT     = arg        jméno výstupního souboru
```

- vytvořte si vlastní format hlavičky, který bude obsahovat všechny nutné údaje pro dešifrování
- 💀 návratové hodnoty: nula pro úspěch, 1 až 127 pro neúspěch.
- 💀 kontrolujte výstupy funkcí (čtení souboru, šifrování, dešifrování…)
- 💀 dešifrovaný soubor MUSÍ být stejný jako původní
- 💀 nezapomínejte, že velikost zašifrovaného klíče je závislá na velikosti klíče asymetrického, tzn. není známá předem
- 💀 nenačítejte celý soubor do paměti
- [-1b] nastane-li chyba, výstupní soubor se nevytvoří (žádný!)
- [+1b] pátý nepovinný poziční parametr pro šifrování:

```
./encrypt e publickey filein fileout CIPHERNAME
```

```
CIPHERNAME  = arg        jméno symetrické šifry ve formátu, jak to bere EVP_get_cipherbyn
```

Předchozí zadání

▼ úkol 1: klasické šifry ve Wolfram Mathematica

1. [0.5] Najděte klíč pro Caesarou šifru k uvedenému v notebooku textu.
2. [1] Najděte klíč pro afinní šifru k textu, který najdete v mém kanálu v Teams v souboru `cv-2--<čas cvičení>`. Tam v záložce `tasks` najdete svůj login a příslušný šifrový text. Soubor vždycky přidám těsně před začátkem cvičení.
3. [0.5] Najděte klíč pro transpoziční šifru k textu, který dostanete od vámi zvoleného kolegy. Najít kolegu můžete např. v mém kanálu v Teams. Pozor, nevolte text s moc jednoduchou faktorizaci délky (dva prvočísla).

▼ úkol 2: částečné prolomení jednosměrnosti hashovací funkce

Jádro úkolu zůstává stejné jako je to na stránce zadání (../../tutorials/05.html#_zadání-úkolu). Dejte pozor na použitou hash funkci.

Moje požadavky:

- binárka se musí jmenovat `hash` a brát jeden a pouze jeden parametr, a to počet nulových **bitů** na začátku.
- přikládejte makefile, který umí zbuildit vaše řešení.
- format výstupu je
  - 💀 na prvním řádku vypište **hexadecimálně bez mezer** obsah zprávy, hash které produkuje požadovaný počet nul na začátku
  - 💀 na druhém řádku hash té zprávy.
- příklad výstupu:

```
$ ./hash 4
3132330a
0ee756a11951fa5d3d9d1c8b85f3e33583569d5360cdd4599a3c62edb9d2cc943b2fd38ddd65af72d95e2d71d
```

- jak ověřit?

```
$ echo -n "3132330a" | xxd -r -ps | openssl sha384
0ee756a11951fa5d3d9d1c8b85f3e33583569d5360cdd4599a3c62edb9d2cc943b2fd38ddd65af72d95e2d71d
# xxd -r -ps na stdout vypíše bajty které mají hexadecimální hodnoty dané na stdin
```

Přehlednější zdroják ze cvičení example_hash.cc (../../media/teacher/shchamar/example_hash.cc)

▼ úkol 3: operační módy blokových šifer

Jádro úkolu zůstává stejné jako je to na stránce zadání (../../tutorials/06.html#_zadání).

Moje požadavky:

- binárka se musí jmenovat `block` a brát 3 poziční parametry v následující formě:

```
./block ACTION MODE FILENAME
```

```
ACTION   = {e,d}          má-li program provést šifrování (encrypt) nebo dešifrování (dec
MODE     = {ecb,cbc,...}  operační mód blokové šifry
FILENAME = arg            soubor k {,de}šifrování
```

- výstup
  - návratová hodnota
    - 0 pokud je všechno v pořádku
    - 1 až 127 pokud něco se nepodařilo
  - pokud `FILENAME = BASENAME.tga`, tak výstup je `BASENAME_MODE_ACTION.tga`
    - například po provedení `./block e cbc pic.tga` výstupní soubor je `pic_cbc_e.tga`
    - když budu dešifrovat soubor `pic_cbc_e.tga` výstupní soubor bude `pic_cbc_e_cbc_d.tga`
- 💀 soubor, který jste dešifrovali musí být na velikost a po bajtech stejný, jako původní soubor
- 💀 klíč a inicializační vektor jsou hardcoded a jsou dostačující délky
  - nechceme překvapení stylu "použití neinicializované paměti", která v každém běhu obsahuje něco jiného než v tom předchozím
- 💀 soubor nenačítejte celý do paměti, ale po nějakých smysluplných blocích, třeba násobky 4096
- [-1b] kontrolujte výstupy funkcí, jsou vaši největší přátele pro debug
- [-1b] kontrolujte, jestli vstup dává smysl (soubor menší než minimální velikost hlavičky,…)
- [-1b] do komentáře k merge requestu napište co vidíte, jaký je rozdíl mezi šifrováním pomocí ECB a CBC