Šifrování souborů

Termín odevzdání:	**31.08.2024 23:59:59**	5663039.538 sec
Hodnocení:	**10.0000**	
Max. hodnocení:	**10.0000** (bez bonusů)	
Odevzdaná řešení:	1 / 20 Volné pokusy + 10 Penalizované pokusy (-10 % penalizace za každé odevzdání)	
Nápovědy:	0 / 5 Volné nápovědy + 5 Penalizované nápovědy (-10 % penalizace za každou nápovědu)	

Vaším úkolem je realizovat dvě (či více) funkcí, (ne celý program), které dokáží zašifrovat a dešifrovat obrázkový soubor ve formátu **TGA**.

Pro naši úlohu budeme uvažovat zjednodušenou formu obrázku:

- Povinná hlavička: 18 bytů - tyto bajty nijak nemodifikujeme, jen je překopírujeme do zašifrovaného obrázku.
- Volitelná část hlavičky: velikost se vypočítá z povinné části hlavičky - tuto část hlavičky budeme považovat za obrázková data, tj. beze změn je **zašifrujeme společně** s obrázkovými daty.
- Obrázková data: zbytek.

Parametry Vámi implementovaných funkcí jsou:

```
bool encrypt_data (const string & in_filename, const string & out_filename, crypto_config & config)
```

- `in_filename` - vstupní jméno souboru,
- `out_filename` - výstupní jméno souboru,
- `config` - datová struktura `crypto_config` popsaná níže.
- Návratová hodnota je `true` v případě úspěchu, `false` v opačném případě. K neúspěchu dochází, pokud je soubor nějakým způsobem nevalidní (schází povinná hlavička, nepodaří se otevřít, číst, zapsat, …) nebo se nepodaří opravit nevalidní konfiguraci `crypto_config`u.

Funkce `decrypt_data` využívá stejného rozhraní, jen provádí inverzní operaci vzhledem k šifrování. Dojde tedy ke zkopírování povinné části hlavičky, která **není** šifrovaná, následně zbytek souboru dešifrujeme stejným způsobem, jako probíhalo šifrování. V tomto případě ale očekáváme předání validního dešifrovacího klíče a IV (pokud je potřeba). Pokud tyto parametry nemáme, nemůžeme data dešifrovat a program by měl zahlásit chybu (`return false`).

Datová struktura `crypto_config` obsahuje:

- zvolenou blokovou šifru zadanou pomocí jejího názvu,
- tajný šifrovací klíč a jeho velikost,
- inicializační vektor (IV) a jeho velikost.

Při šifrování může dojít k následujícímu problému: pokud je šifrovací klíč (či IV) nedostačující, (tedy jejich délka není alespoň tak velká, jakou požaduje zvolená bloková šifra nebo úplně chybí), musí dojít k jejich bezpečnému vygenerování. Pokud zadaná bloková šifra IV nepotřebuje (a nemusí Vám tedy být předaný), nový IV negenerujte! Nezapomeňte případně vygenerované klíče a IV uložit do předávané struktury configu!

Následující funkce pro šifrování se Vám budou hodit:

- `EVP_EncryptInit_ex`, resp. `EVP_DecryptInit_ex`,
- `EVP_EncryptUpdate`, resp. `EVP_DecryptUpdate`,
- `EVP_EncryptFinal_ex`, resp. `EVP_DecryptFinal_ex`.

V dokumentaci openssl se můžete podívat, jaké další funkce byste mohli (a měli) využít. Hint: Neexistuje nějaká obecnější funkce, která by tyto funkce zastřešovala?

Ve výchozím stavu mají blokové šifry zapnuté zarovnání (padding) a proto délka výsledného zašifrovaného souboru může být větší než původní. To je chtěné (a v testech očekávané chování) a **neměli** byste ho měnit.

V testovacím prostředí budete omezeni nejen časem, ale i velikostí dostupné paměti, Váš program může být případně násilně ukončen. Pokuste se proto zbytečně nevyužívat haldu nebo ji ideálně vůbec nepoužívejte. V 90 % případů si vystačíte pouze se zásobníkem.

Odevzdávejte zdrojový soubor, který obsahuje implementaci požadované funkce `encrypt_data` a `decrypt_data`. Do zdrojového souboru si můžete přidat i další Vaše podpůrné funkce, které jsou z `encrypt_data` a `decrypt_data` volané. Funkce bude volána z testovacího prostředí, je proto důležité přesně dodržet zadané rozhraní funkce.

Za základ pro implementaci použijte kód z přiloženého archivu níže. Ukázka obsahuje testovací funkci `main`, uvedené hodnoty jsou použité při základním testu. Všimněte si, že vkládání hlavičkových souborů a funkce `main` jsou zabalené v bloku podmíněného překladu (`#ifdef/#endif`). Prosím, ponechte bloky podmíněného překladu i v odevzdávaném zdrojovém souboru. Podmíněný překlad Vám zjednoduší práci. Při kompilaci na Vašem počítači můžete program normálně spouštět a testovat. Při kompilaci na Progtestu funkce `main` a vkládání hlavičkových souborů „zmizí“, tedy nebude kolidovat s hlavičkovými soubory a funkcí `main` testovacího prostředí.

Poznámky:

- **POZOR!** Odevzdaná úloha na Progtestu není zárukou splněné úlohy! Více informací se dozvíte od svého cvičícího.

- Pečlivě ošetřujte souborové operace. Testovací prostředí úmyslně testuje Vaši implementaci pro soubory neexistující, nečitelné nebo soubory s nesprávným datovým obsahem.
- Při implementaci lze použít C i C++ rozhraní pro práci se soubory, volba je na Vás. Stejně tak lze využít struktury STL.
- V zadání úlohy jsou ukázky běhu. V přiloženém archivu najdete sadu testovacích souborů a jim odpovídajících ekvivalentů šifrovaných nějakou blokovou šifrou.
- Vstupní a výstupní soubory mohou být velké, větší než je velikost dostupné paměti. Obecně se proto při práci se soubory snažíme data zpracovávat průběžně. Není rozumné celý vstupní soubor načíst do paměti a pak jej v paměti zpracovávat. Poslední test kontroluje paměťové nároky Vašeho řešení. Selže, pokud se pokusíte udržovat v paměti najednou celé soubory nebo jejich velké části.
- Nezapomeňte, že i při selhání nějaké dílčí funkce se musíte správně postarat o dynamicky získané prostředky.
- Šifrovací klíč a IV jsou obecně „nějaká“ data, pracovat s nimi tedy jako s ASCII řetězci je cesta ke zkáze (viz cvičení).
- Šifrování jako takové je deterministické, generování klíčů by ale nemělo. Testovací prostředí kontroluje správnost šifrovací funkce tím, že Vámi zašifrovaný soubor zkusí dešifrovat a výsledek porovná s očekáváním (dochází tedy k binární shodě s originálním souborem).
- Pro snazší implementaci obsahuje struktura `crypto_config` chytré ukazatele (`std::unique_ptr`), pro jejich vytváření je vhodné použít funkci `std::make_unique`. Pokud budete potřebovat někde předat holý ukazatel, použijte třídní metodu `get()`.

Vzorová data:		**Download**
Odevzdat:	Choose File No file chosen	**Odevzdat**

☐ **Referenční řešení**

1	**14.04.2024 17:47:20**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	10.0000	

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test se soubory z ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.007 s (limit: 5.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Nespravne vstupy': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.005 s (limit: 4.993 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Nahodny test - sifrovani': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.938 s (limit: 4.988 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Nahodny test - desifrovani': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.592 s (limit: 4.050 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test osetreni I/O chyb': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.205 s (limit: 3.458 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test pametove narocnosti': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.516 s (limit: 10.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi hodnotami + test prace s pameti - sifrovani': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.978 s (limit: 9.484 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi hodnotami + test prace s pameti - desifrovani': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.512 s (limit: 8.506 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Celkové hodnocení: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00)
- Celkové procentní hodnocení: 100.00 %
- Celkem bodů: 1.00 * 10.00 = 10.00

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**14**	--	--	--
Řádek kódu:	**187**	**13.36 ± 12.18**	**45**	`main`
Cyklomatická složitost:	**60**	**4.29 ± 4.27**	**17**	`main`