Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Knihovna pro off-line práci se šifrovanými kontejnery TrueCryptu

Petr Mück

Vedoucí práce: Ing. Josef Kokeš

11. května 2015

Poděkování

Rád bych tímto poděkoval svému vedoucímu, Ing. Josefu Kokešovi, za veškerou pomoc při tvorbě této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2015 .....

Odkaz na tuto práci

Mück, Petr. *Knihovna pro off-line práci se šifrovanými kontejnery TrueCryptu.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato práce se zabývá návrhem a implementací knihovny pro off-line prohlížení a extrakci souborů šifrovaných kontejnerů TrueCryptu. Práce popisuje TrueCrypt a jeho funkce, souborové systémy FAT, NTFS, ext a některé knihovny pro jejich implementaci a realizaci knihovny, její strukturu a možné rozšíření. Výsledkem práce je funkční knihovna v programovacím jazyku C pro operační systém Linux navržena tak, aby byla snadno rozšiřitelná.

Klíčová slova truecrypt, knihovna, C, extrakce souborů, prohlížení složek, AES, Twofish, Serpent, FAT, NTFS, ext

Abstract

This thesis describes the design and implementation of library for off-line browsing and extraction of TrueCrypt containers. The thesis describes TrueCrypt and its functions, FAT, NTFS and ext filesystems and some of the libraries supporting them and realization of the library, its structure and possible expansion. The result is a C library compatible with Linux OSes, designed to be easily expanded.

Keywords truecrypt, library, C, file extraction, folder browsing, AES, Twofish, Serpent, FAT, NTFS, ext

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Šifrování dat je velmi důležité pro bezpečnost jakékoliv práce na počítači, o to více s dnešním rozšířením internetu a různých online aplikací, například cloudových služeb. I běžný uživatel může mít zájem na zabezpečení svých dat na disku. Jednou z aplikací, které mu toto umožňují je TrueCrypt.

TrueCrypt je již starší, nadále nevyvíjený program umožňující vytváření souborového systému uvnitř heslovaného souboru. Ten pak lze přes TrueCrypt se správným heslem připojit, což umožňuje práci s jeho soubory stejným způsobem jako s klasickou diskovou jednotkou, například USB flash diskem. Výhodou je jeho přívětivé uživatelské rozhraní vhodné i pro laika. Program ukončil vývoj začátkem roku 2014, navázaly na něj ale nově vytvořené programy nabízející stejné či velmi podobné funkce, podporující i starší kontejnery vytvořené TrueCryptem, například VeraCrypt či CipherShed.

Problém nastane v situaci, kdy nemáme možnost kontejner na zařízení připojit. Zařízení nemusí souborový systém podporovat či neumožňuje jeho připojení. Je také možné, že o připojení souboru nemáme zájem z jiných důvodů, například jde o součást jiného již existujícího programu či o zpracování většího množství kontejnerů zároveň. Dalším důvodem může být například viditelnost daného kontejneru ze všech procesů daného operačního systému (případ Microsoft Windows) nebo nutnost speciálních oprávnění pro jeho připojení (administrátorská práva na Microsoft Windows).

Tato bakalářská práce se zabývá právě návrhem a implementací knihovny, která by umožňovala práci s kontejnery bez nutnosti jejich předchozího připojení. Knihovna by měla být snadno rozšiřitelná o další souborové systémy či algoritmy. Funkce knihovny je také třeba předvést na demonstračním programu.

První kapitolou této bakalářské práce je popis TrueCryptu samotného. Tato část je rozdělena na dvě části, jednu zabývající se faktickými informacemi o TrueCryptu a druhou popisující jeho vnitřní fungování, například použité šifry či způsob získávání klíčů ke kontejnerům. Správná implementace této části je klíčová pro dešifrování dat obsažených v kontejneru.

Druhou kapitolou je stručný popis souborových systémů FAT, NTFS, ext a některých knihoven implementujících práci s nimi. Znát strukturu souborového systému je podstatné pro zpracování datové části TrueCrypt kontejneru.

Poslední kapitolou je pak samotná realizace knihovny. Obsahuje podrobnější popis struktury knihovny, jak je možně knihovnu dále rozšiřit a některých nedostatků se stručným návrhem, jak by je bylo možné řešit. Dále obsahuje krátký popis demonstračního programu sloužícího k předvedení základní práce s knihovnou.

Kapitola 1

TrueCrypt - principy, historie

TrueCrypt je bezplatný nástroj umožňující šifrování obsahu disku. Je dostupný pro většinu běžných operačních systémů a nabízí přívětivé uživatelské rozhraní. Využívá tzv. OTFE (on-the-fly-encryption, šifrování „za běhu“), která umožňuje uživateli přístup k šifrovaným datům stejný jako k nešifrovaným a šifrovací operace provádí automaticky. Zdrojové kódy jsou volně dostupné, ale používá vlastní licenci, která není běžně považována za open source.[2]

1.1 Historie a vývoj

TrueCrypt je založen na starším, již nepodporovaném šifrovacím nástroji E4M (Encryption for the Masses, „šifrování pro každého“) na Microsoft Windows. Krátce po vydání verze 1.0 přerušili autoři kvůli sporům ohledně autorských práv ke zdrojovým kódům E4M na několik měsíců vývoj a distrubuci programu.[3]

Během vývoje došlo k mnoha změnám a zlepšením uživatelského rozhraní (například změna hesla nebo možnost nastavení oblíbených svazků pro načtení při startu systému), podpoře pro více systémů (Linux ve verzi 4.0, Mac OS ve verzi 5.0) i celkovým možnostem šifrování (kaskádové šifry, použití klíčových souborů). Pravděpodobně nejvíc změn se ale dočkaly používané šifrovací algoritmy a jejich operační módy.

V původní verzi byly k dispozici šifrovací algoritmy IDEA, Blowfish, CAST5 a TripleDES. Postupně byly všechny odstraněny (IDEA z licenčních důvodů, zbytek z důvodů odstranění podpory pro šifry operující na 64bitových blocích) a nahrazeny finalisty soutěže AES (Advanced Encryption Standard) – samotným vítězem AES (Rijndael) a algoritmy Serpent a Twofish. Operační módy jednotlivých algoritmů se postupně změnily z CBC[1] přes LRW[2] až konečně na XTS[3]. Taktéž se změnily používané hashovací funkce – z původně jediné SHA-1 na SHA-512, RIPEMD-160 a Whirlpool. [4]

[1] Cipher Block Chaining
[2] Liskov, Rivest, and Wagner.
[3] XEX-based tweaked-codebook mode with ciphertext stealing

1.2 Možnosti

Základní funkcionalitou nástroje TrueCrypt je vytváření souborů, v kterých je část disku šifrována a přístupná pouze zadáním správného hesla. Postupem času ale TrueCrypt začal nabízet řadu dalších možností, jak ještě zlepšit zabezpečení dat či dát uživateli větší pohodlí a kontrolu nad obsahem.

Jednou z těchto možností je přidání jednoho či více klíčových souborů (keyfile), jejichž obsahem je pak ještě modifikováno původní heslo. K získání přístupu k datům tedy nestačí mít pouze text hesla, ale i všechny klíčové soubory v původním stavu . Je také možné si zvolit, jakou šifru, případně kombinaci více šifer, bude TrueCrypt používat nebo také jaký hashovací algoritmus použije pro vytvoření klíčů z hesla. Další možností je zašifrovat již existující souborový systém místo vytváření nového nebo na systémech Windows zašifrovat kompletní systémový disk s tím, že uživatel bude muset zadat heslo při každém spuštění počítače.

Za zvláštní zmínku stojí možnost vytvoření tzv. hidden volume (skrytého svazku), kdy v samotném TrueCrypt kontejneru existují dva souborové systémy přístupné pomocí různých hesel. Skrytý souborový systém je od obyčejného rozdílný v tom, že ze struktury šifrovaných dat v kontejneru nelze usoudit, zda existuje nebo ne – jak hlavička skrytého svazku, tak jeho obsah je v případě jeho neexistence naplněn náhodnými daty. Také je možné v situaci, kde je uživatel nucen vydat heslo k zašifrovanému kontejneru sdělit heslo od obyčejného svazku a data na skrytém svazku zůstanou soukromá. Pro systémy Windows lze takto skrýt i celý operační systém v rámci šifrování celého systémového disku.

1.3 Ukončení podpory

Autoři nástroje TrueCrypt (TrueCrypt Foundation) vydali 28. května 2014 oznámení o ukončení veškeré podpory a vývoje, doprovázené vydáním poslední verze 7.2, která už nepodporuje vytváření nových TrueCrypt kontejnerů, pouze jejich čtení. Zároveň stáhli většinu obsahu z oficiálních stránek (www.truecrypt.org) a vyvěsili zde upozornění, že nástroj nemusí již nadále být bezpečný, spolu s výzvou k přechodu na jiný podobný program.

TrueCrypt, navzdory doporučení autorů, zůstal i po skončení vývoje populárním šifrovacím nástrojem. Důvodem mohou být přívětivé uživatelské rozhraní, bezpečnost nabízených šifrovacích algoritmů nebo popularizace TrueCryptu díky soudním sporům, kde obžalovaný nebyl soudem donucen svůj TrueCryptem zašifrovaný disk odemknout a forensním expertům se nepodařilo svazek prolomit.[5] Těsně před skončením podpory proběhla první fáze nezávislého bezpečnostního auditu TrueCryptu – Open Crypto Audit Project. Ta obsahovala analýzu zdrojové kódu TrueCryptu a nebylo při ní nalezeno žádné velké bezpečnostní riziko ani žádný backdoor v programu. [6] Druhá část ana-

lýzy proběhla počátkem roku 2015 a odhalila několik nedostatků, které by za určitých okolností mohly snižovat bezpečnost TrueCryptu. Jedním z nich je možná chyba při práci s generátorem náhodných čísel Microsoft Windows, kdy TrueCrypt přijal vygenované data i při chybě generátoru. Tato chyba ale není kritická, jelikož TrueCrypt pro generování náhodných čísel používá i jiné zdroje (například pohyb myší). Dalším je například možnost zneužití dočasné paměti (cache) použité u šifrování pomocí AES, k tomu je ale ovšem potřeba přístup ke stejnému stroji v průběhu vytváření svazku. Z auditu ve výsledku vyplynuly určitá varování pro uživatele a vývojáře navazující na TrueCrypt, samotnou funkci a důvěryhodnost programu to ale příliš nezpochybnilo. [7]

Na dotaz, zda zvažují změnit licenci na open source pro možnost navázání na projekt TrueCrypt, odpověděli autoři negativně s důvodem, že si nemyslí, že by měl být projekt dále větven.[8] I přesto vznikly nadále vyvíjené alternativy, které z velké části vycházejí právě z posledních verzí nástroje TrueCrypt, například CipherShed nebo VeraCrypt.

Kapitola 2

TrueCrypt - realizace

Zdrojové kódy TrueCryptu jsou veřejně přístupné. Kromě toho byly veškeré použité metody detailně popsány na oficiálních stránkách projektu, než byly, spolu s veškerým dalším obsahem, staženy v květnu 2014 v souvislosti s ukončením vývoje nástroje. Samotný proces šifrování a dešifrování je tedy veřejně známý, stejně jako struktura kontejneru a algoritmy používané k vytváření šifrovacích klíčů.

2.1 Struktura kontejneru

Kontejner TrueCryptu se skládá z dvou základních částí – hlavičky (header) a datové části. Kontejner může navíc obsahovat skrytý oddíl – jeho vlastní hlavičku a datovou část – přístupný pomocí vlastního hesla a tedy i klíčů. Důležitou vlastností kontejneru je, že je při vytváření naplněn náhodnými daty a až na 64 bajtů náhodně generované soli na začátku hlavičky je celý zašifrován. To má za důsledek, že před dešifrováním souboru nelze spolehlivě poznat, že jde o TrueCrypt kontejner.[4]

2.1.1 Hlavička a datová část

Na začátku každého TrueCrypt kontejneru je 128 kB rezervováno jako prostor pro hlavičky (header). Hlavní hlavička je obsažena v prvních 512 bajtech TrueCrypt kontejneru, skrytá hlavička má také délku 512 bajtů a nachází se v půlce rezervované části (odsazená 64 bajtů od začátku). Prvních 64 bajtů hlavičky tvoří nešifrovaná hašovací sůl (salt), která je použita pří vytváření šifrovacích klíčů.[5] Kombinací této soli a uživatelem zadaného hesla se vytvoří klíč k hlavičce (header key), s pomocí kterého je poté dešifrován zbytek hlavičky.

[4] Náhodná data jsou využívána například některými programy pro bezpečné mazání souborů

[5] Sůl je náhodně generovaná, není třeba ji šifrovat – pro útočníka působí stejně náhodně jako zbytek souboru, neprozradí tedy TrueCryptem šifrovaný soubor

Dešifrovaná hlavička obsahuje důležitá data týkající se daného kontejneru. První z nich je slovo TRUE v ASCII, které je obsaženo na začátku dešifrované oblasti hlavičky a s pomocí něhož TrueCrypt pozná, zda uživatelem zadané heslo bylo správné či ne. Pravděpodobně nejdůležitejším prvkem hlavičky jsou šifrovací klíče hlavní datové části (master key). Pomocí nich pak probíhá šifrování a dešifrování samotných dat souborového systému v rámci TrueCrypt kontejneru. Dále se zde nachází kontrolní součty hlavičky a klíčů, adresa začátku, velikost datové části a další informace. Hlavička zahrnuje také informaci o velikosti skrytého oddílu, ale tato informace je pro hlavní hlavičku vždy rovná 0, ať skrytý oddíl existuje nebo ne. V případě skryté hlavičky skutečně obsahuje velikost skrytého oddílu.

Datová část tvoří převážnou část TrueCrypt kontejneru. Obsahuje souborový systém šifrovaný pomocí hlavního šifrovacího klíče získaného z hlavičky kontejneru a stejného šifrovacího algoritmu, jaký byl použit pro šifrování hlavičky. Při tvorbě nového TrueCrypt kontejneru je obsah datové části naplněn náhodnými daty. [9]

2.1.2 Skrytý oddíl

Kontejner může obsahovat také skrytý oddíl – druhý souborový systém v rámci jednoho kontejneru. Ten je přístupný pomocí odlišného hesla a nesdílí s hlavním oddílem žádná data. Zahrnutí skrytého oddílu je při tvorbě kontejneru volitelné. Pokud skrytý oddíl v kontejneru existuje, obsahuje vlastni hlavičku zašifrovanou vlastním klíčem. Má tedy vlastní heslo, odlišné od hesla hlavního a i vlastní sůl, nešifrovanou na začátku skryté hlavičky. Součástí hlavičky skrytého oddílu je i velikost skryté datové části – údaj vždy rovný 0 u hlavní hlavičky kontejneru. Pokud kontejner skrytý oddíl neobsahuje, je oblast skryté hlavičky naplněna náhodnými daty. [9]

2.1.3 Záložní hlavičky

TrueCrypt kontejner obsahuje na konci dvě záložní hlavičky, jednu pro hlavičku hlavního oddílu a druhou pro hlavičku oddílu skrytého. Každá z nich používá vlastní klíč - stejné heslo jako předloha každé z nich, ale jinou sůl. V případě poškození hlavičky se pak TrueCrypt pokusí použít její zálohu k obnově - hlavičky jsou znovu zašifrovány s použitím nové soli. K obnovení lze použít i externí zálohu hlavičky, kterou nabízí TrueCrypt ve svém uživatelském rozhraní.[10]

2.2 Klíče

TrueCrypt používá pro šifrování dva typy klíčů. Jeden, vytvořený s pomocí hesla a náhodně generované soli, je použit k šifrování hlavičky. Sůl je poté uložena na začátku hlavičky v nešifrované podobě a s její pomocí je stejný klíč

znovu vygenerován pro účely dešifrování.[11] Druhý, náhodně vygenerovaný TrueCryptem samotným, je uložen na konci zašifrované hlavičky a je použit k šifrování datové části kontejneru. Délka obou klíčů je závislá na použitém šifrovacím algoritmu a operačním módu. Od verze 5.0 TrueCrypt pro veškeré šifrování používá operační mód vyžadující primární a sekundární klíč délky 256 bitů, tedy 512 bitů celkem.

Pro vytváření hlavičkového klíče TrueCrypt používá algoritmus PBKDF2-HMAC [6], která pomocí hesla, soli, zvoleného hashovacího algoritmu a počtu iterací vytvoří klíč požadované délky. TrueCrypt používá 2000 iterací pro RIPEMD-160 a 1000 pro SHA-512 a Whirlpool. Význam většího počtu iterací je zejména ve zvýšení času nutného ke generování jednoho klíče a tím větší ochraně proti brute-force útokům [7]. Průběh algoritmu PBKDF2 popisuje algoritmus 1. [12] [13]

TrueCrypt umožňuje uživateli zvolit, jakou hashovací funkci použije pro generování klíče k hlavičce.

Algorithm 1 PBKDF2

function F(PRF, Password, Salt, count, i)
    $U_1 \leftarrow$ *PRF (Password, Salt || INT (i))*
    for $a = 2 \rightarrow count$ **do**
        $U_a = PRF(Password, U_{a-1})$
    $Result = U_1 \oplus U_2 \oplus ... \oplus U_{count}$
    return *Result*

function PBKDF2(PRF, Password, Salt, count, keyLen)
    blockCnt $\leftarrow$ *Ceil (keyLen / blockLen)*
    extraBytes $\leftarrow$ *keyLen - (blockCnt - 1) * blockLen*

    for $i = 1 \rightarrow blockCnt$ **do**
        $T_i = F(PRF, Password, Salt, count, i)$
    $KEY = T_1 \parallel T_2 \parallel ... \parallel T_{blockCnt<0..extraBytes-1>}$
    return *KEY*

PRF – Pseudorandom function, pseudonáhodná funkce, například hashovací algoritmus
$INT(i)$) – Číslo bloku i převedeno na 32bitový integer

2.2.1 Hashovací funkce

Účel hashovacích funkcí je vytvořit pro libovolně velký vstup výstup fixní délky. TrueCrypt využívá hashovací funkce v rámci vytváření hlavičkového

[6] Password-based key derivation function 2, Hash-based message authentication code

[7] Útok hrubou silou, spočívá v generování všech možných kombinací dokud nenajde správné heslo

klíče s pomocí hesla, soli a případně klíčových souborů (keyfile). Hlavními vlastnostmi hashovacích funkcí jsou: [14]

1. libovolná velikost vstupu,
2. fixní délka výstupu,
3. jednosměrnost - z výstupu funkce je nemožné získat její vstup,
4. ochrana proti kolizím prvního řádu - nalezení libovolného páru dvou vstupů, které produkují stejný výstup,
5. ochrana proti kolizím druhého řádu - nalezení druhého vstupu k fixnímu prvnímu, který produkuje stejný výstup.

2.2.1.1 Padding

Před samotným spuštěním hashovacího algoritmu je nutné provést tzv. padding - doplnění vstupních dat tak, aby byly násobkem velikosti vstupního bloku. Padding probíhá následovně:

1. Spočte se velikost vstupních dat.
2. Na konec vstupu se přidá bit 1 a poté tolik bitů 0, aby výsledná délka vstupu byla násobek délky vstupního bloku hashovací funkce.
3. Na závěr se místo doplněných nulových bitů na konci vstupních dat přidá velikost vstupních dat.

Délka vstupního bloku RIPEMD-160 a Whirlpool je 512 bitů, SHA-512 je 1024 bitů. Počet bitů tvořící informaci o délce na konci vstupních dat RIPEMD-160 je 64 bitů, SHA-512 128 bitů a Whirlpool 256 bitů. [14]

2.2.1.2 RIPEMD-160

RIPEMD-160 je hashovací funkce se 160 bitovým výstupem. Byla vyvinuta a publikována otevřenou vědeckou komunitou v Evropě, oproti široce rozšířeným hashovacím funkci z rodiny SHA, které byly vyvinuty NSA [8] v relativní tajnosti.[15] Mezi ostatními hashovacími funkcemi TrueCryptu vyniká odlišnou délkou výsledného hashe, která je vynahrazena větším počtem iterací při vytváření hlavičkového klíče.

Jak už bylo výše řečeno, RIPEMD-160 používá 160 bitové bloky výstupu, rozdělené do 5 32 bitových bloků. V hashovací funkci figuruje na počátku inicializační vektor délky 160 bitů jako výchozí hash. Na ten jsou postupně po blocích aplikována vstupní data. Pro každý blok vstupních dat se předchozí hash zduplikuje, paralelně se na každý aplikuje půlka (16 bitů) vstupního

[8] National Security Agency – Národní bezpečnostní agentura, americká vládní organizace

bloku a proběhne 5 rund po 16 krocích. Na závěr se oba výsledky modulárně sečtou, což dá výsledný hash. Ten pak tvoří vstup pro další vstupní blok. Konečný výsledek tvoří poslední výstupní hash. Při aplikaci jednoho bloku vstupu na předchozí hash probíhají bitové operace and, xor, or, not, bitová rotace a modulární sčítání. [16] [17]

2.2.1.3 SHA-512

SHA (Secure Hash Algorithm) je rodina standardizovaných hashovacích funkcí publikovaná NIST. [9] Do té dále patří skupiny algoritmů SHA-0 až SHA-3 s tím, že SHA-512 patří do skupiny SHA-2. Číslo 512 značí velikost výstupního bloku algoritmu, která je 512 bitů. V minulosti TrueCrypt používal i starší verzi standardu, SHA-1, s velikostí výstupního bloku 160 bitů. Oba algoritmy jsou si ale do značné míry podobné.

Upravená vstupní data jsou poté rozdělena po blocích na 16 64 bitových slov. V hashovací funkci figuruje na počátku inicializační vektor délky 512 bitů jako výchozí hash. Na ten jsou postupně aplikována vstupní data tak, že pro jeden blok vstupních dat proběhne celkem 80 rund bitových operací a výstupem je nový 512 bitový hash. Ten pak tvoří vstup pro další blok vstupních dat. Výsledným hashem je poslední takto vzniklý hash. Při aplikaci jednoho bloku vstupu na předchozí hash probíhají bitové operace and, xor, or, bitová rotace, logický posun doprava a modulární sčítání. [14] [15]

2.2.1.4 Whirlpool

Whirlpool je hashovací funkce s 512 bitovým výstupem. Její vnitřní struktura je velmi podobná šifře AES (Rijndael), s kterou sdílí jednoho z autorů. Podrobnější popis šifry AES se nachází dále v práci.

Vstupní data jsou funkcí vnitřně vnímána jako bajty - skupiny osmi bitů. Jeden vstupní blok velikosti 512 bitů je tedy rozdělen na 64 bajtů, které jsou nadále zapsány do matice o rozměrech 8x8 bajtů. V hashovací funkci figuruje na počátku inicializační vektor obsahující 512 nulových bitů jako výchozí hash, z kterého se derivují rundové klíče. Pro každý vstupní blok 512 bitů je provedeno 10 rund. Každá runda pak obsahuje 4 operace - AddRoundKey, MixRow, ShiftColumn a SubBytes [10]. Výstupem těchto 10 rund aplikovaných na jedne vstupní blok je nový hash, který je poté použit k derivaci klíče pro další vstupní blok. Výsledným hashem je poslední takto vzniklý hash transformovaný z matice zpět na řetězec bitů. [18] [19]

[9] National Institute of Standards and Technology – Národní institut standardů a technologie, americká laboratoř

[10] AddRoundKey – zakomponuje klíč pro současnou rundu, MixRow – zamíchá bajty v každém řádku matice, ShiftColumn – posune prvky ve sloupcích matice, SubBytes – funkce aplikovaná na každý bajt zvlášť permutující bity

2.3 Šifrovací algoritmy

TrueCrypt během svého vývoje prošel řadou změn týkajících se používaných šifrovacích algoritmů a operačních módů. Od verze 5.0 až do ukončení vývoje zůstaly algoritmy AES, Serpent a TwoFish, všechno finalisté soutěže AES - Advanced Encryption Standard. V soutěži šlo o nalezení bezpečné a zároveň výkonné symetrické blokové šifry, která měla nahradit předchozí standardní šifru DES. TrueCrypt podporuje také tzv. kaskádové šifry - dva nebo tři šifrovací algoritmy použité po sobě, s různými na sobě nezávislými klíči. Všechny použité algoritmy používají 256 bitové klíče a šifrují bloky o délce 128 bitů. [20] V minulosti byly využívány šifry IDEA, TripleDES, CAST5 a Blowfish.[4] Všechny kontejnery vytvořené od verze 5.0 dále využívaly pouze operační mód XTS, TrueCrypt ale i nadále podporoval otevírání starších kontejnerů šifrovaných pomocí algoritmů v módech CBC a LRW. [4]

2.3.1 Operační módy šifer

Blokové šifry operují samy o sobě jen s jedním blokem dat a klíčem. Pro šifrování větších bloků dat pak existuje řada různých operačních módů. Operační mód působí jako doplněk blokové šifry, kde nějakým způsobem využívá postupně bloky vstupního textu (plaintext) pro generování bloků šifrového textu (ciphertext). Velikost bloku je závislá na použité blokové šifře. U většiny operačních módů je také používan inicializační vektor, který zajišťuje, že dva shodné vstupní texty mohou být zašifrované stejným klíčem s různým inicializačním vektorem a vznikne jiný šifrový text. Různé operační módy se liší ve využití otevřeného textu, v aplikaci inicializačního vektoru a šifrovací nebo dešifrovací funkce či v zacházení s nekompletními bloky. Vlastnosti šifrového textu se také liší, například propagací chyby při změně dat. [21]

TrueCrypt v minulosti využíval operační módy CBC[11] a LRW[12]. Ve verzi 5.0 pak přešel na XTS[13] a využíval jej až do ukončení vývoje. Módům CBC a LRW se zde více věnovat nebudu, pro více informací o těchto operačních módech doporučuji například [22] [23].

2.3.1.1 XTS

Operační mód XTS je operační mód XEX [14] doplněný o operaci CTS (Ciphertext stealing).

CTS umožňuje šifrovat vstupní text nedělitelný velikostí šifrového bloku bez změny délky výsledného šifrového textu. Probíhá tak, že před zašifrováním poslední zkrácené části vstupního textu se doplní do velikosti bloku pomocí

[11] Cipher Block Chaining
[12] Liskov, Rivest, and Wagner.
[13] XEX-based tweaked-codebook mode with ciphertext stealing
[14] Xor-encrypt-xor

bitů z předposledního bloku šifrového textu. Tento blok je poté zašifrován a prohozen s předposledním blokem. Poslední blok je pak oříznut do původní velikosti vstupního textu. Dešifrování probíhá inverzně - začne se předposledním blokem, poslední bity dešifrovaného bloku doplní poslední částečný blok. Ten je dešifrován, bloky jsou prohozeny a délka oříznuta na původní. [24]

Kromě CTS je průběh módu identický s XEX, jehož průběh popisuje algoritmus 2. [25]

Algorithm 2 XEX

function XEX($E_K, Plaintext, sectorNumber, blockNumber$)
$X \leftarrow E_K(sectorNumber) \times \alpha^{blockNumber}$
$C \leftarrow E_K(Plaintext \oplus X) \oplus X$
return C

E_K – Encryption algorithm with key K, algoritmus E používající šifrovací klíč K
α – Primitivná element tělesa $GF(2_128)$ určeného polynomem $x_128 + x_7 + x_2 + x + 1$

2.3.2 AES

AES [15] je bloková šifra s velikostí bloku 128 bitů a klíče 128, 192 nebo 256 bitů. Jde o výherce stejnojmenné soutěže o novou šifru, která měla na místě standardu nahradit předchozí DES [16].

Blok délky 128 bitů je chápán jako matice 4x4 bajtů. V závislosti na délce klíče pak AES provádí určitý počet rund - 10 pro 128bitový, 12 pro 192bitový a 14 pro 256bitový klíč. V každé rundě aplikuje na matici určité bitové operace. Pro šifrování jednoho bloku jsou z použitého klíče generovány 128 bitů dlouhé podklíče. Jejich počet je rovný počtu rund zvýšenému o jedna a jsou aplikovány jako xor právě šifrovaného bloku dat před první rundou, mezi rundami a po poslední rundě. Algoritmem generování podklíčů (tzv. key schedule) se zde blíže zabývat nebudu, pro bližší popis doporučuji například [26] .

Pro každý 128bitový blok vstupního textu rozděleného do matice 4x4 bajtů probíhají tyto operace:

1. KeyExpansion - vytvoření podklíčů užívaných při šifrování bloku
2. Před začátkem rundových operací se jednou aplikuje AddRoundKey - xor všech bitů šifrovaného bloku jedním z vytvořených podklíčů
3. Průběh jednotlivých rund
   a) SubBytes - operace aplikovaná na každý bajt zvlášť, substituje bity
   b) ShiftRows - posunuje bajty v jednotlivých řádcích matice doleva

[15] Advanced Encryption Standard
[16] Data Encryption Standard

c) MixColumns - bajty v jednotlivých sloupcích matice jsou zkombinovány

d) AddRoundKey

4. Poslední runda (neobsahuje operaci MixColumns)

a) SubBytes

b) ShiftRows

c) AddRoundKey

Průběh operací rundy AES je znázorněn na obrázku 2.1. Při dešifrování probíhají inverzní operace v opačném pořadí.[27]

2.3.3 Serpent

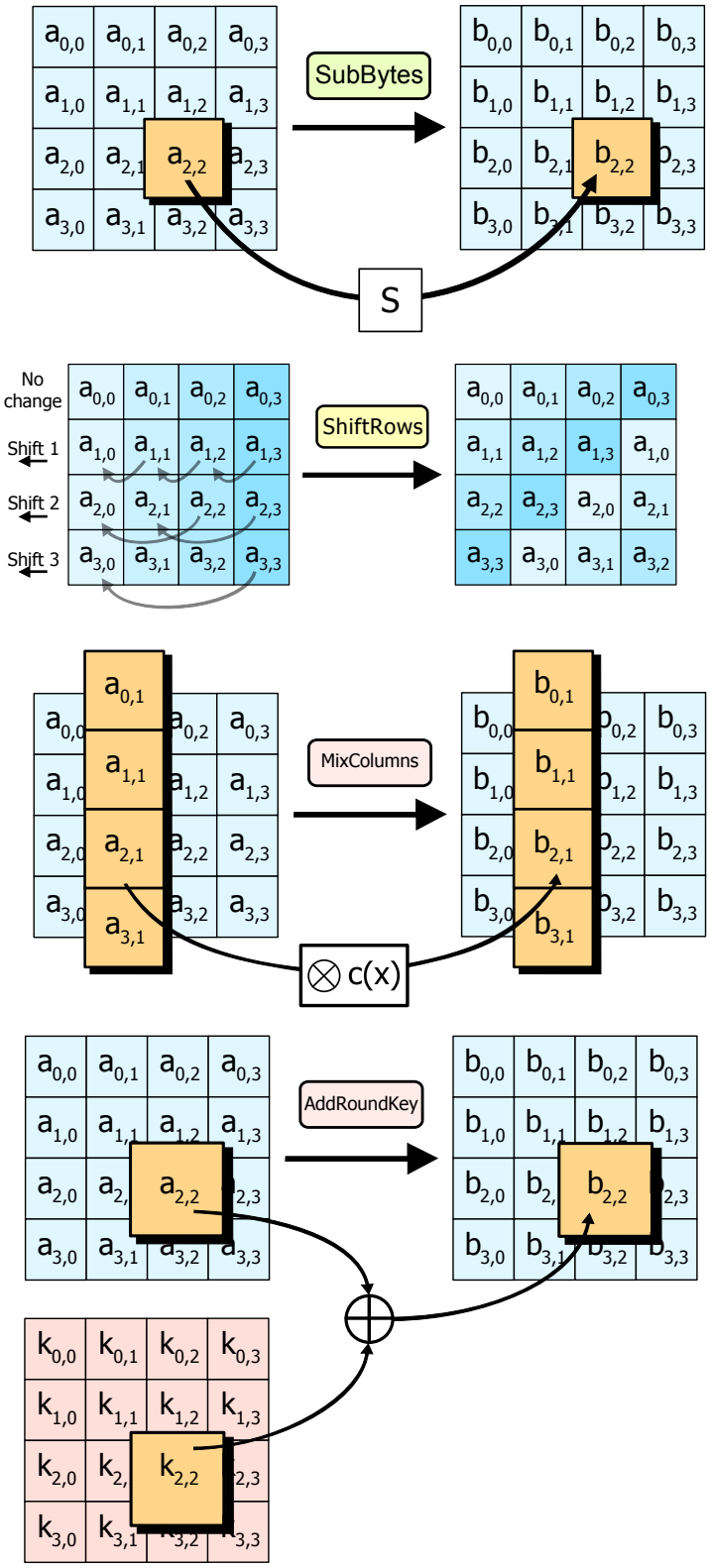
Serpent je bloková šifra, která se stala jedním z finalistů soutěže AES. Velikost bloku má 128 bitů a velikost klíče libovolnou do 256 bitů (zbytek do 256 je dolněn jedním bitem 1 a nulami), ale běžně je fixován na hodnotách 128, 192 a 256 bitů.

Blok délky 128 bitů je chápán jako 4 32 bitová slova. Na něj je aplikována počáteční permutace bitů, poté 32 rund bitových operací a koncová permutace bitů. Z klíče se získá 33 podklíčů dlouhých 128 bitů, ty jsou pak využívané v rundách k bitové operaci xor na vstupní 128 bitový blok. V rundách dále probíhají permutační operace a lineární transformace dat. Výše popsaná šifra AES používá principielně velmi podobné operace. Šifrou Serpent se zde více zabývat nebudu, pro další informace doporučuji například [28] [29] .

2.3.4 Twofish

Twofish je bloková šifra a další z finalistů soutěže AES. Velikost bloku je 128 bitů a klíčů 128, 192 či 256 bitů. Částečně vychází z dříve používané šifry Blowfish stejného autora.

Twofish používá polovinu svého klíče pro generování podklíčů a druhou pro generování tzv. S-boxů používaných v jednotlivých rundách. Celkový počet rund je 16. Twofish rozdělí blok 128 bitů vstupních dat nejprve na 4 slova délky 32 bitů, ty poté na 4 bajty, na které aplikuje dané S-boxy. Výsledky jsou poté mezi slovy kombinovány, lineárně transformovány a jsou na ně aplikovány podklíče. Pro další rundu se pořadí prvního a druhého páru slov prohodí. Šifrou Twofish se zde více zabývat nebudu, pro další informace doporučuji například zde [30] [31] .



Obrázek 2.1: Jednotlivé operace rundy AES [1]

Kapitola 3

Souborové systémy

TrueCrypt podporu souborových systémů přenechává operačnímu systému, proto možnosti vytváření či načítání určitých TrueCrypt svazků závisí na použitém operačním systému. Tuto metodu ovšem nemohu použít, protože by to porušilo požadavek na offline zpracování kontejneru, je tedy nutné použít externí knihovny, pokud možno nezávislé na operačním systému.

3.1 FAT

FAT[17] je třída souborových systémů vyvinutá firmou Microsoft. Patří do ní souborové systémy FAT8(dnes již nepoužívaný), FAT12, FAT16, FAT32 a exFAT, novější souborový systém určený především pro flash disky. Čísla v označení FAT značí počtu bitů použitých k adresaci jednotlivých alokačních jednotek (clusterů).

Dnes již FAT v operačních systémech nahradily nové, pokročilejší souborové systémy (například NTFS ve Windows), nadále jsou však podporovány prakticky všemi zařízeními z důvodů kompatibility. Hlavními problémy FAT jsou omezení co se týče velikosti jak celého souborového systému, tak jednoho souboru a celkového počtu souborů v systémů. [32]

Existuje řada knihoven implementující přístup k FAT souborovým systémům, zejména kvůli vestavěným (embedded) systémům a přenosným zařízením (například USB flash disky) které je často využívají. Dalším důvodem je, že struktura FAT systémů je už delší dobu známá a vytvoření knihovny proto není náročné.

[17] File Allocation Table

3.1.1 Dostupné knihovny

3.1.1.1 FatFs

FatFs je knihovna podporující souborové systémy FAT12/16/32. Je nezávislá na platformě. Nabízí možnost přístupu k souborům a složkám pomocí standardního C rozhraní, podporuje dlouhé názvy a neomezený počet zároveň používaných souborových systémů. Pro použití je nutné implementovat nízkoúrovňové funkce specifické pro cílové zařízení, na oficiálních webových stránkách jsou k dispozici ukázkové implementace pro některé z častěji používaných zařízení. Na stránkách je k dispozici i méně náročná verze knihovny Petit FatFs vhodná pro zařízení s velmi omezenou pamětí a výkonem. Knihovna je open-source.[33]

3.1.1.2 Ultra-Embedded FAT16/32 File System Library

Ultra-Embedded FAT16/32 File System Library je, jak už název napovídá, knihovna podporující souborové systémy FAT16/32 se zaměřením pro použití na vestavěných zařízeních. Je nezávislá na platformě. Nabízí přístup k souborům a složkám pomocí standardního C rozhraní, podporuje dlouhé názvy souborů a je málo náročná na hardware, zejména na paměť. Pro použití je nutné implementovat funkce pro čtení a zápis paměťových bloků specifické pro dané zařízení. Nevýhodou knihovny je, že nepodporuje více otevřených souborových systémů zároveň. Knihovna je open source, autoři ale nabízí i licenci vhodnou pro projekty s uzavřeným zdrojovým kódem.[34]

3.1.1.3 smxFS

smxFS je knihovna podporující souborové systémy FAT12/16/32 od firmy Micro Digital Inc. Je nezávislá na platformě. Nabízí přístup k souborům a složkám pomocí standardního C rozhraní, podporu dlouhých názvů souborů, neomezeného počtu otevřených souborových systémů zároveň a jiné. Součástí jsou již implementované ovladače pro řadu běžných zařízení pro které lze knihovnu přímo použít bez jakýchkoliv úprav. Pro použití na jiných zařízeních je potřeba implementovat 7 funkcí specifických pro dané zařízení. Knihovna je vydávána pod komerční licencí.[35]

3.1.1.4 RTFiles-32

RTFiles-32 je knihovna od firmy On Time podporující souborové systémy FAT12/16/32, exFAT a ISO 9660. Je nezávislá na platformě. Obsahuje své vlastní rozhraní i s mnoha pokročilými funkcemi, podporu dlouhých názvů souborů i neomezený počet zároveň otevřených souborových systémů. Knihovna obsahuje kompletní ovladače pro velké množství zařízení včetně příkladů, jak přidat nové. K přidání dalšího podporovaného zařízení jsou potřeba implementovat 3 funkce. Knihovna je vydávána pod komerční licencí.[36]

3.2 NTFS

NTFS[18] je souborový systém vyvinutý firmou Microsoft. Jde o nástupce starších FAT souborových systémů od Microsoftu a je používán jako preferovaný souborový systém na všech jejich operačních systémech od Windows NT 3.1 dále.

NTFS přinesl mnoho změn oproti staršímu FAT. Některými z nich jsou například větší podporované velikosti disku a jednotlivých souborů, šifrování a omezení přístupu ke konkrétním souborům, komprese souborů či schopnost samoopravovat chyby na disku. Také používá skupinu specifických souborů za začátku souborového systému pro uchovávání metadat ohledně všech souborů a složek, což pak automaticky využívá například k zálohování systémových souborů.

Knihoven implementujících NTFS ovladače je méně než těch implementujících FAT zejména z toho důvodu, že přesná vnitřní struktura NTFS je prozatím veřejnosti neznámá.[37] Nevýhodou pro naše účely je, že u obou zde zmíněných knihoven jde o kompletní ovladače, tedy úprava čtecí funkce pro zahrnutí šifrování nemusí být tak jednoduchá jako u většiny knihoven pro FAT a bude nejspíše vyžadovat větší změny. Pravděpodobně by bylo nutné ovladač také upravit či použít jen část, aby splňoval podmínku pro offline zpracování kontejneru.

3.2.1 Dostupné knihovny

3.2.1.1 NTFS-3G

NTFS-3G je multiplatformní ovladač pro podporu NTFS souborových systémů. Je použitelný na mnoha operačních systémech, například Linux, FreeBSD, OpenSolaris, Haiku a dalších. Podporuje operace čtení i zápisu, kompresi souborů, úpravu přístupových práv, částečně žurnálování (journaling) s možností pozdější obnovy a jiné. Nepodporuje čtení či zápis souborů šifrovaných pomocí NTFS. Pro méně výkonná zařízení má problémy s výkonem. Ovladač je open-source, ale navazuje na něj komerční verze Tuxera NTFS Embedded, která nabízí stejné možnosti s lepším výkonem a je vhodná například pro vestavěná zařízení.[38]

3.2.1.2 Paragon Software - NTFS for Linux

NTFS for Linux je ovladač pro podporu NTFS souborových systémů firmy Paragon Software. Podporuje čtení i zápis a nabízí řadu utilit pro změnu a kontrolu údajů souborového systému, například defragmentaci souborového systému nebo kontrolu celistvosti souborů. Firma nabízí ovladače jak pro vestavěná zařízení, tak pro stolní počítače. U vestavěných zařízení je rychlost

[18] New Technology File System

srovnatelná s nativními systémy a požadavky na hardware jsou velmi malé. Knihovna je vydávána pod komerční licencí.[39]

3.3 ext

ext[19] je třída souborových systémů nativní pro operační systém Linux. Patří do ní souborové systémy ext, ext2, ext3 a ext4 s tím, že ext byl prakticky okamžitě nahrazen pokročilejším ext2.

ext2 je stále upřednostňovaným souborovým systémem pro SD karty nebo USB flash disky, protože neobsahuje žurnálování (journaling), který zvyšuje počet operací nutných pro každý zápis a to snižuje výkon. Naopak, jeho následovník ext3 již žurnálování má, což umožňuje napravení chyb při havárii systému a v důsledku zvyšuje spolehlivost. ext3 je výchozím souborovým systémem mnoha Linuxových distribucí. ext4 nabídl mnoho novinek oproti ext3, například větší podporované velikosti souborového systému nebo nový algoritmus alokování volných bloků pro nové soubory. Je také zpětně kompatibilní a lze v něm používat starší ext2 a ext3 souborové systémy beze změny s tím, že mohou využívat některé z vylepšených vlastnosti ext4.

Vzhledem k nativní podpoře ext souborových systémů na Linuxu se externí ovladače pro zápis a čtení ext souborových systémů omezují na Windows a OS X. Podobně jako u NTFS jde o kompletní ovladače, jejich implementace bude tedy nejspíše obtížnější než implementace FAT souborových systémů, které nabízejí snadno upravitelné čtečí funkce. Pravděpodobně by bylo nutné ovladač také upravit či použít jen část, aby splňoval podmínku pro offline zpracování kontejneru.

3.3.1 Dostupné knihovny

3.3.1.1 Ext2Fsd

Ext2Fsd je ovladač pro Microsoft Windows (2000, XP, Vista a Windows 7), který podporuje práci s ext2/3/4 souborovými systémy. Podporuje operace čtení a zápisu pro ext2/3, dále pak například částečnou podporu žurnálování a read-only podporu pro nově přidané blokové schéma ext4 (tzv. extent). Ovladač je vydáván pod GPL licencí. [40]

3.3.1.2 Paragon Software - ExtFS for Windows

ExtFS for Windows je ovladač pro podporu ext souborových systémů firmy Paragon Software. Je funkční pro všechny verze Windows novější než XP. Plně podporuje čtení a zápis z ext2/3/4 souborových systémů a nabízí i několik pomocných utilit k jejich spravování, například vytváření či úpravě existujících diskových oddílů. Ovladač je ke stažení zdarma pro osobní použití s omezenými

[19] Extended File System

možnostmi (například menší počet utilit), plná verze je pak vydávána pod komerční licencí. [41] Paragon Software vydala stejný driver i pro operační systém Mac OS X.[42]

3.4 Srovnání knihoven

Jednotlivé knihovny i souborové systémy se v mnohém liší. Nabízejí různé způsoby implementace pro programátory a různé dodatečné utility pro uživatele. Pro cíl této práce většina těchto prvků není až tak důležitá. Co může hrát roli je množství možných otevřených souborových systémů zároveň a podpora dlouhých názvů souborů.

Většina knihoven si je v těchto oblastech rovnocenná, jak je vidět z tabulky 3.1. Pro další implementaci bude tedy záležet převážně na preferenci programátora a typu licence.

Název knihovny	Soub. systém	Dlouhé názvy	Více souborů	Licence
FatFs	FAT	x	x	open source
Ultra-Embedded FS	FAT	x		open source
smxFS	FAT	x	x	komerční
RTFiles-32	FAT	x	x	komerční
NTFS-3G	NTFS	x	x	open source
Paragon NTFS	NTFS	x	x	komerční
Ext2Fsd	Ext2/3/4	x	x	open source
Paragon ExtFS	Ext2/3/4	x	x	komerční

Tabulka 3.1: Srovnání knihoven implementujících souborové systémy

Kapitola 4

Realizace

Cílem práce bylo vytvořit knihovnu, která bude umožňovat off-line práci s TrueCrypt kontejnery. Pracovní název mnou vytvořené knihovny řešící tento problém je tc-lib (TrueCrypt-library).

Implementace knihovny je v jazyce C. Ke kompilaci lze použít například standardní Linuxový C kompilátor *gcc*, kde je potřeba zahrnout všechny soubory knihovny s koncovkou „.c“. Jako příklad kompilace může posloužit kompilační skript *compile* zahrnutý v knihovně, který je použitý pro demonstrační program. Jazyk C jsem zvolil zejména z důvodu, že se mi v něm pracuje pohodlně, existuje pro něj řada knihoven a je podporovaný většinou operačních systémů.

Knihovna je implementovaná pro operační systém Linux. Důvodem mojí volby je především větší pohodlí pro programovaní, lze ale i předpokládat, že podobná knihovna bude populární spíše u uživatelů Linuxových systémů právě kvůli větší popularitě mezi programátory. Knihovna ale, vyjma externích knihoven, nepoužívá žádné prvky specifické pro operační systém Linux. Vytvoření verze pro jiný operační systém by tedy nemělo být příliš náročné.

Implementace počítá s Little Endian pořadím bajtů. Pro všechny TrueCryptem podporované systémy je toto standard, nemělo by to tedy způsobovat větší problémy.

4.1 Použité knihovny

V předchozí kapitole jsem nastínil některé z ovladačů, které lze použít k implementaci jednotlivých souborových systémů v tc-lib. Možností je více a při implementaci dalších je vhodné zvážit jejich klady a zápory, například rychlost, celkové možnosti či omezení nebo portabilita na různé operační systémy.

4.1.1 FAT16/32

Pro implementaci souborových systémů FAT16 a FAT32 jsem zvolil knihovnu Ultra-Embedded FAT16/32 File System Library. Důvodem mojí volby byla především velmi snadná implementace, která spočívá v implementaci pouhých dvou funkcí (čtení a zápis bloků), nízké hardwarové nároky knihovny vhodné například i pro vestavěná zařízení a rozhraní velmi podobné rozhraní používané standardními C funkcemi pro práci se soubory.

Až v pozdější fázi implementace jsem narazil na omezení této knihovny, kterým je možnost otevřít pouze jeden souborový systém zároveň a pevně daná definice funkcí pro čtení a zápis. V konečné verzi jsem tuto knihovnu nechal i s těmito nedostatky. Při dalším vývoji bych ale doporučil knihovnu nahradit jinou, která podobné omezení nemá, například populární FatFs.

4.1.2 Kryptografické funkce

Pro implementaci funkce PBKDF2-HMAC, s pomocí které je generován hlavičkový klíč, hashovacích algoritmů (RIPEMD-160, SHA-512, Whirlpool) a šifrovací funkce AES jsem zvolil knihovnu OpenSSL. Důvody jsou její popularita, jednoduchost použití vysoko úrovňových funkcí, existence verzí pro většinu populárních operačních systémů a v neposlední řadě má dřívější zkušenost s prací s OpenSSL.

Pro implementaci šifrovací funkce Twofish jsem využil zdrojové kódy TrueCryptu. Důvodem je zejména málo dalších alternativ, které obsahují jak samotný Twofish, tak implementaci XTS operačního módu. Výhodou je možné snadnější přidání šifrovací funkce Serpent, která je v TrueCryptu také obsažena a velkou část implementace má s Twofishem společnou. Nevýhodou je složitější struktura vypůjčeného zdrojového kódu, který není příliš dobře čitelný.

4.2 Struktura knihovny

tc-lib je členěný do tří základních vrstev. Uživatelská vrstva je nejvyšší a nabízí funkce pro přímé použití knihovny. tc-lib vrstva provádí operace v rámci knihovny, volané uživatelskou vrstvou a volá externí funkce. Nejnižší vrstvou je vrstva externích knihoven, která obsahuje části nutné pro funkci tc-lib vypůjčené z externích zdrojů.

4.2.1 Uživatelská vrstva

Nejvyšší vrstvu knihovny tc-lib tvoří uživatelská vrstva. Ta obsahuje uživatelsky přívětivé rozhraní, které tvoří základ knihovny a jejich vstup a výstup je evidentní. Pro pouhou implementaci knihovny v cílovém programu stačí správně použít funkce této vrstvy. Vrstva obsahuje pět základních funkcí -

uložení hlavičkových údajů do struktury, inicializace použitého souborového systému, uložení obsahu konkrétní složky do pole struktur, extrakce konkrétního souboru z kontejneru do externího souboru a ukončení práce s kontejnerem.

4.2.1.1 Uložení hlavičkových údajů

Při práci s libovolným TrueCrypt kontejnerem prvním krokem musí být dešifrování hlavičky. S použitím zadaného hesla a soli připojené na začátku hlavičky v nešifrované podobě získáme hlavičkový klíč. S jeho pomocí poté dešifrujeme obsah hlavičky a tím získáme klíčové údaje o daném kontejneru. Důležité je pamatovat, že neznáme použitý šifrovací algoritmus ani hashovací funkci, proto je nutné vyzkoušet všechny možnosti. Správnou hlavičku TrueCrypt (a stejně tak tc-lib) pozná podle ASCII řetězce TRUE na začátku dešifrované hlavičky a platných kontrolních součtů hlavičky. Šifra používaná k šifrování hlavičky je vždy použita i pro šifrování samotné datové části.

Tuto roli plní v knihovně tc-lib funkce *decrypt_header*, jejíž implementace se nachází v souboru *tclib-header*. Jejím vstupem je soubor otevřený pro čtení, heslo a instance struktury *TCHeader*, do které mají být zapsány výsledné údaje o hlavičce. Návratovou hodnotu funkce pak je hodnota 1 pro úspěšné získání hlavičky, hodnota 0 pro neúspěch - platná TrueCrypt hlavička nebyla nalezena.

Funkce prochází dvěma cykly pro každý typ šifry (jednoduché, dvojité a trojité kaskády). Vnější cyklus generuje s užitím různých hashovacích algoritmů a funkce PBKDF2-HMAC klíče různých délek s pomocí hesla a soli hlavní či skryté hlavičky. Pro každý takto vytvořený klíč probíhá pak vnitřní cyklus, který dešifruje hlavičku pomocí vytvořeného klíče všemi šiframi daného typu a kontroluje výsledek.

Pokud funkce najde v dešifrované hlavičce řetězec TRUE a platné kontrolní součty, naplní strukturu TCHeader údaji z hlavičky a ukončí se. Struktura *TCHeader* obsahuje jednotlivé položky dešifrované hlavičky jako pole bajtů. Kromě toho obsahuje informace o tom, zda získaná hlavička je nebo není hlavičkou skrytého svazku a typ použité šifry. Přesný obsah struktury *TCHeader* naleznete v souboru *tclib-includes*.

4.2.1.2 Inicializace souborového systému

Značná část knihoven implementující práci se souborovými systémy vyžaduje její inicializaci před samotným použitím. Tuto roli v rámci knihovny plní funkce *init_fs* obsažená v souboru *tclib-filesystem*. Jejím vstupem jsou struktura *TCHeader* obsahující údaje o dešifrované hlavičce kontejneru a externí cesta ke kontejneru, se kterým pracujeme.

Knihovny implementující souborové systémy se od sebe v mnohém liší a je tedy pravděpodobné, že i jejich inicializační funkce se mohou výrazně lišit. Pro

inicializaci ovšem knihovny nemohou potřebovat více specifických informací než ty, které předáváme, protože struktura *TCHeader* obsahuje veškeré údaje potřebné k dešifrování souborového systému.

Funkce má na starosti pouze volání funkcí nižší úrovně (vrstvy tc-lib) v závislosti na typu zadaného souborového systému. Pro podrobnější popis těchto funkcí referujte na následující sekci zabývající se tc-lib vrstvou knihovny.

4.2.1.3 Uložení obsahu složky

Jednou z klíčových funkcí práce s jakýmkoliv souborovým systémem je vypsání obsahu složky. V závislosti na souborovém systému pak o každém souboru či složce můžene získat určité údaje. Běžnými údaji jsou například poslední čas změny či velikost souboru. Pro uživatele implementující knihovnu může být ale vhodné nevypisovat všechny soubory se všemi atributy, ale vybírat jen některé z nich, nebo je například podle nějakého atributu řadit. Z tohoto důvodu jsem zvolil implementaci funkce jako uložení obsahu složky do pole struktur, které uchovávají informace o každém prvku složky a zbytek by už neměl být pro uživatele velký problém implementovat.

Uložení obsahu složky v tc-lib má na starosti funkce *get_directory* obsažená v souboru *tclib-filesystem*. Jejím vstupem je typ souborového systému (podporovaného tc-lib[20]), cesta k žádané složce v rámci kontejneru a proměnná, do které je uložen počet souborů ve složce. Návratovou hodnotou je pak ukazatel na začátek pole struktur *file_info*. Funkce alokuje potřebné pole dynamicky podle potřeby, je tedy nutné pamět později opět uvolnit jakmile není potřeba.

Struktura *file_info* obsahuje většinu běžných atributů souborů a složek, které by uživatel mohl chtít použít. Jednotlivé údaje dostupné na určitém souborovém systému se mohou lišit, proto je nutné pro každý implementovaný souborový systém přidat nové potřebné položky do struktury *file_info* a napsat potřebnou parsovací funkci, která strukturu získanými údaji naplní. Přesný obsah struktury *file_info* naleznete v souboru *tclib-includes*.

Funkce má na starosti pouze volání funkcí nižší úrovně (vrstvy tc-lib) v závislosti na typu zadaného souborového systému. Pro podrobnější popis těchto funkcí referujte na následující sekci zabývající se tc-lib vrstvou knihovny.

4.2.1.4 Extrakce souboru

Další základní funkcí tc-lib je extrakce souboru z kontejneru do externího souboru. Protože funkce přečtení souboru je nutností každé knihovny implementující souborový systém, struktura funkcí použitých v této části bude pravděpodobně velmi jednoduchá.

Extrakce souboru je v knihovně implementována ve funkci *extract_file* obsažené v souboru *tclib-filesystem*. Jejím vstupem je typ souborového systému

[20] v době psaní této práce pouze souborový systém FAT16/32

(podporovaného tc-lib[21]), cesta k žádanému souboru k extrakci v rámci kontejneru a cesta k externímu souboru, do kterého budou data extrahována. Návratovou hodnotu funkce pak je hodnota 1 pro úspěšnou extrakci, hodnota 0 pro neúspěšnou s chybou vypsanou na chybový výstup.

Funkce má na starosti pouze volání funkcí nižší úrovně (vrstvy tc-lib) v závislosti na typu zadaného souborového systému. Pro podrobnější popis těchto funkcí referujte na následující sekci zabývající se tc-lib vrstvou knihovny.

4.2.1.5 Ukončení práce s kontejnerem

Po ukončení práce s TrueCrypt kontejnerem je vhodné bezpečně odstranit hlavičkové údaje z paměti, protože obsahují citlivé údaje potřebné pro jejich dešifrování (zejména klíče) a případně uzavřít otevřený souborový systém je-li to třeba.

Roli odstranění hlavičky v knihovně plní funkce *delete_header* souboru *tclib-header*. Funkce dostane parametrem ukazatel na strukturu *TCHeader*, kterou funkce přepíše v paměti nulami. Struktura funkce je velmi jednoduchá a ne příliš zajímavá, proto se jí zde více věnovat nebudu.

Ukončení práce se souborovým systémem má na starosti funkce *close_fs* souboru *tclib-filesystem*. Funkce dostane parametrem ukazatel na strukturu TCHeader uchovávající údaje o načteném svazku a cestu k otevřenému TrueCrypt kontejneru, který uzavře. Funkce pouze volá funkce specifické pro jednotlivé souborové systémy, ty se mohou svou strukturou v mnohém lišit. Předané údaje však jednoznačně identifikují, o kterou instanci souborového systému jde, i pokud by byly zároveň otevřeny hlavní i skrytý svazek stejného kontejneru zároveň. Jsou tedy pro ukončení práce s jakýmkoliv souborovým systémem dostatečné.

4.2.2 tc-lib vrstva

Střední vrstvu knihovny tvoří funkce spojující nejnižší vrstvu používající externí knihovny a nejvyšší uživatelskou vrstvu. Funkce této vrstvy je třeba doplnit při implementaci dalších souborových systémů či šifrovacích algoritmů.

tc-lib vrstva obsahuje veškeré vnitřní funkce knihovny tc-lib. V této sekci tyto funkce rozdělím do tří tematických částí - šifrovací, souborová a pomocná část. Šifrovací a souborové části jdou snadno rozdělit podle souborů, v kterých se nalézá jejich implementace. Pomocná část pak obsahuje veškeré zbývající funkce, těmi se zde do větší hloubky zabývat nebudu.

Tato vrstva obsahuje množství funkcí specifických pro určitou šifru či souborový systém. Jejich realizace by však měla být mezi různými knihovnami velmi podobná, minimálně co se týče vstupů a výstupů, a proto se zde budu zabývat podrobněji jen realizací šifry AES a souborového systému FAT16/32.

[21] viz předchozí poznámka

4.2.2.1 Dešifrování bloků

Knihovna tc-lib ze svojí podstaty provádí velký počet dešifrovacích operací, ať už jde o dešifrování hlavičky či datové části. Pro tyto potřeby je vhodné mít všechny dostupné šifrovací funkce na jednotném místě s velmi podobnou definicí.

Tuto roli v knihovně plní funkce *tclib_decryptBlocks* implementována v souboru *tclib-crypto*. Funkce má za parametry použitou šifru, ukazatele vstupní a výstupní pole bajtů, číslo, počet a velikost sektorů k dešifrování a použitý klíč. Funkce ale samotné dešifrování neprovádí, pouze volá specifické dešifrovací funkce umístěné v patřičných souborech, například *AES_decryptBlocks* souboru *tclib-aes*.

Důležité je, že číslo sektoru neurčuje sektor k dešifrování v rámci vstupního pole bajtů, ale určuje, jaké číslo má sektor, kterým vstup začíná. Například pokud chceme dešifrovat 3 bloky dat od sektoru číslo 120, najdeme si začátek sektoru číslo 120 ve vstupním souboru a ukazatel na něj dáme do parametru *in*. Pak jej předáme s ostatními parametry funkci. Důvodem, proč je třeba šifrovací funkci předávat i číslo sektoru je, že se z něj odvozuje tweak klíč, který je potřeba při dešifrování v operačním módu XTS.

Dalším specifikem je nakládání s klíčem. Funkci je předán ukazatel na klíč jako celek, ovšem v závislosti na použité šifře je potřeba s ním různě operovat. Pro jednoduché šifry tvoří výsledný šifrovací klíč pouze prvních 64 bajtů, kde prvních 32 bajtů je hlavní a zbylých 32 tweakovací klíč, avšak pro kaskádové šifry je už situace složitější. Například pro kaskádovou šifru AES-Twofish je potřeba provést dešifrování nejdříve AES, poté Twofishem s dvěma rozdílými klíči, které ale nejsou v souvislých blocích. Jejich pořadí v rámci 128 bajtového bloku je hlavní Twofish, hlavní AES, tweakovací Twofish a tweakovací AES klíč.[43] Při implementaci dalších šifrovacích algoritmů je nutné být ohledně tohoto velmi obezřetný.

4.2.2.2 Dešifrování bloků - AES

Obecně to, co žádáme po jednotlivých šifrovacích algoritmech je získat dešifrovaný vstupní text ve výstupním poli bajtů. Jedním z implementovaných šifrovacích algoritmů je AES, k jehož realizaci jsem použil funkce knihovny OpenSSL.

Funkce *AES_decryptBlocks* se nachází v souboru *tclib-aes*. Jejími parametry jsou ukazatel na vstupní a výstupní pole bajtů, číslo a velikost sektoru a dvě části XTS klíče - hlavní a tweakovací. Vstupní parametry zůstávají kromě hodnot ve výstupním bajtovém poli nedotčeny.

Funkce začíná spočtením čísla tweaku, který je nutný k dešifrování. Tweak je přímo závislý na čísle sektoru v rámci celého souboru a je uchováván jako 16 bajtové pole. K tomuto je využita pomocná funkce *calculateTweak* implementovana taktéž v souboru *tclib-aes*.

Dalším krokem je „složení“ výsledného dešifrovacího klíče. OpenSSL má tu vlastnost, že klíč pro XTS operační mód nemá externě rozdělený v parametrech na dvě části, hlavní a tweakovací, ale očekává klíč dvojnásobné délky, kde první polovinu tvoří klíč hlavní a druhou tweakovací. Například pro šifru AES 256 v XTS očekává klíč délky 512 bitů, kde na pozicích 0-255 je hlavní klíč a 256-511 tweakovací. Toto může způsobit problémy v situacích, kde klíče nejsou tvořeny souvislými bloky dat, například pro kaskádové šifry TrueCryptu. Proto je nutné klíč předat v parametrech po částech a poté složit v rámci funkce. Po skončení funkce se paměť se složeným klíčem opět uvolní. Různé knihovny s klíčem nakládají různě a toto skádání klíčů v jiných případech vůbec nemusí být potřeba, ale vyřešit tyto nesrovnalosti je přesně ten důvod, proč zde tyto funkce spojující uživatelskou a externí vrstvu jsou.

Po těchto dvou operacích probíhá už samotná dešifrovací rutina OpenSSL - inicializace funkcí a kontextu, dešifrování, jeho finalizace a opět uvolnění a úklid OpenSSL prostředků.

4.2.2.3 Uložení obsahu složky do pole struktur - FAT16/32

Uložení složky do pole struktur *file_info* je voláno funkcí uživatelské vrstvy *get_directory*. Výpis složky v nějaké formě je poměrně standartní částí knihoven implementující souborové systémy, ale v knihovně tc-lib je tato funkce implementována pomocí vlastní struktury *file_info* použité k uložení údajů o jednotlivých souborech a složkách vypisované složky.

Funkce *get_directory_fat* souboru *tclib-fat* má dva vstupní parametry. Prvním je cesta ke složce v rámci TrueCrypt kontejneru, jejíž obsah chceme uložit a číslo, v kterém funkce vrací počet prvků dané složky. Návratovou hodnotou funkce je pak ukazatel na počátek pole struktur *file_info*.

Průběh funkce je velmi podobný funkci *fl_listdirectory* obsažené v použité zdrojové knihovně Ultra-Embedded. Nevýhodou této funkce bylo, že obsah složek rovnou vypisovala na standartní výstup, což není příliš praktické. Pro uživatele, ´který má zájem implementovat tuto funkci, je mnohem užitečnější uložit obsah složky, kde s jejími atributy pak může nějakým způsobem manipulovat, například řadit podle různých atributů či některé při výpisu úplně vynechat. Samotný výpis pole na výstup už je poté triviální.

Funkce *fl_listdirectory*, kterou jsem k implementaci *get_directory_fat* použil, používá převážně nízko-úrovňové funkce knihovny Ultra-Embedded. Není tedy příliš čitelná. Funkce začíná kontrolou, zda je souborový systém inicializován, případně jej funkce inicializuje. Následně se zamkne přístup k souborovému systému a otevře složka k výpisu. Poté probíhá cyklus čtení složky, kde jsou informace o prvcích složky ukládány do struktury *fs_dir_ent*, z které jsou pak okamžitě vypisovány na výstup. Jakmile už není co číst, složka se uzavře a odemkne se přístup k souborovému systému.

Pro potřeby tc-lib zůstala funkce z převážné části stejná, jediným rozdílem je, co se provádí ve čtecím cyklu. Funkce prvně projde složkou a jen spočítá

prvky a alokuje pole struktur *file_info* k uložení obsahu složky. Následně projde složku znovu, načte informace o každém prvku do struktury *fs_dir_ent* a tu pak pomocnou funkcí *parse_file_info_fat* převádí na strukturu *file_info*, kterou plní pole. Na konci funkce pouze vrátí ukazatel na začátek tohoto pole v návratové hodnotě a počet prvků pole v druhém parametru funkce.

4.2.2.4 Extrakce souboru - FAT16/32

Extrakce souboru ze souborového systému FAT16/32 je volána funkcí uživatelské vrstvy *extract_file*. Pro extrakci souboru jsou použity funkce externí knihovny *fl_fopen* a *fl_fread*, které fungují velmi podobně standardním C funkcim *fopen* a *fread*.

Samotná funkce *extract_file_fat* souboru *tclib-fat* má jako vstupní parametry cestu k extrahovanému souboru v rámci kontejneru a cestu k externímu souboru, do kterého soubor exportovat. Tělo funkce je v principu velmi jednoduché, především díky uživatelsky přívětivému rozhraní použité knihovny Ultra-Embedded.

V knihovně je tato funkce implementována s volitelnou velikostí zapisovaného bloku. Ta hraje roli především pro velké soubory, kde by mohl být problém celý soubor načíst do paměti před samotným zápisem do externího souboru. Naopak pro soubory či systémy, kde podobné omezení nehrozí a je potřeba extrahovat velký počet souborů, by menší bloky mohly zbytečně snižovat výkon kvůli přidaným instrukcím. Pro určení velikosti zapisovaného bloku je použita konstanta *EXTRACTED_BLOCK_SIZE* definována v souboru *tclib-includes*. Její výchozí hodnota je nastavena na 2048 bajtů. Funkce pak opakovaně čte blok této velikosti a zapisuje jej do výsledného souboru dokud nedojde na konec, kde výsledný i zdrojový soubor opět uzavře.

4.2.2.5 Pomocné funkce

Ne všechny funkce implementované v knihovně tc-lib jsou natolik zajímavé, aby si zasloužily vlastní sekci. Řada z nich ale je pro funkci knihovny klíčová, i přesto že hraje spíše podpůrnou roli. Ve zkratce se tu tedy k těmto funkcím vrátím, ve stejném pořadí jak jsem zmínil funkce, které je využívají.

První pomocné funkce jsou obsažené v samostatném souboru *tc-lib-misc*. Jde o funkci *load_encrypted_header*, která načte a rozdělí hlavičku (hlavní či skrytou v závislosti na tom, na jaké pozici je ukazatel souboru) na nešifrovanou sůl a zašifrovaný zbytek hlavičky. *check_magic* pak kontroluje, zda sedí ASCII řetězec TRUE na začátku dešifrované hlavičky a kontrolní součty. Nakonec *parse_header* rozdělí a uloží do struktury jednotlivé položky TrueCrypt hlavičky.

Pro šifrovací algoritmus AES bylo třeba implementovat jednoduchou funkci *calculateTweak*, umístěnou v souboru *tc-aes*. Ta jednoduše přepočte číslo sektoru na 16 bajtové pole, které je poté použito při dešifrování. Pro šifrovací

funkci Twofish existuje mnoho funkcí, které volá, jsou implementované ve stejném souboru *tclib-twofish*. Tyto funkce jsou ovšem přejaté z původní TrueCrypt implementace a jsou umístěny v jednom souboru jen proto, aby nebylo potřeba zahrnovat v knihovně velké části nepoužívaných funkcí TrueCryptu a byly na jednotném místě. Z tohoto důvodu se jimi zde podrobněji zabývat nebudu.

parse_file_info_fat převádí výstup knihovny Ultra-Embedded při výpisu složky do struktury *file_info* používané knihovnou tc-lib. Variace této funkce budou pravděpodobně potřeba pro každý nově implementovaný souborový systém. Struktura této funkce je velmi jednoduchá a není potřeba ji zde podrobněji rozebírat.

4.2.3 Vrstva externích knihoven

Nejnižší vrstvu pak tvoří funkce vypůjčené z externích knihoven. Tyto funkce jsou volané tc-lib vrstvou, která připravuje data do požadovaného tvaru. Funkce této vrstvy by neměly být měněny, pokud to není nezbytně nutné. Při implementaci nových šifrovacích algoritmů či souborových systémů do této vrstvy budou spadat nově využívané knihovní funkce.

Vrtva obsahuje nejnižší úroveň funkcí knihovny tc-lib. Jde například o funkce dešifrování určitého bloku bajtů danou šifrovací funkcí, vygenerování klíče pomocí algoritmu PBKDF2-HMAC, zisk obsahu konkrétní složky či přečtení konkrétního souboru daného souborového systému.

Funkcemi této vrstvy se zde podrobněji zabývat nebudu, pro jejich podrobnější popis či příklady použití doporučuji oficiální dokumentaci použitých knihoven.

4.3 Rozšíření knihovny

Knihovna tc-lib je navržena tak, aby bylo její rozšíření o další šifrovací algoritmy nebo souborové systémy co nejjednodušší. V této sekci nastíním, co vše je potřeba udělat pro jejich implementaci, jaké soubory je třeba upravit a které funkce přidat či doplnit.

4.3.1 Šifrovací algoritmus

tc-lib v současné formě podporuje pouze algoritmy AES, Twofish a jejich kaskádovou šifru. Při přidání dalších šifrovacích algoritmů - Serpent či některý ze starších šifrovacích algoritmů dříve využívaných TrueCryptem - je potřeba kromě zahrnutí nových externích knihoven upravit prakticky jen tři místa v knihovně. Pro šifru Serpent a nově vzniklé kaskády jsou tyto náležitosti již připravené, stačí pouze přidat jejich implementaci.

4.3.1.1 Hlavičkový soubor *tclib-includes*

Prvním místem, kde je potřeba dělat úpravy, je hlavičkový soubor *tclib-includes.h*, ve kterém se nalézá definice většiny konstant a struktur využívaných knihovnou tc-lib. Jedním z nich je funkce *enum* (enumerate, očíslovat), která přidělí každému jejímu prvku číslo, které jej reprezentuje a prvek je poté nahrazen v kódu na všech místech číslem. Do této funkce je pouze potřeba přidat řetězec reprezentující nově přidanou šifru, který programátor bude používat pro její označení v kódu.

V tomtéž souboru je vhodné upravit celkový počet šifer a jejich typů (jednoduchých, dvojitých a trojitých kaskád). Počty jsou uchovány jako konstanty *CIPHER_COUNT* pro celkový počet šifer a *CIPHER_COUNT* s předponami simple, double a triple pro počet jednoduchých šifer, dvojitých a trojitých kaskád. Počet šifer je pak využíván ve funkci *decrypt_header* souboru *tclib-header*, která je všechny postupně testuje při dešifrovaní hlavičky.

Úpravy tohoto souboru nejsou potřeba při přidání implementace šifry Serpent a jejich kaskád, konstanty jsou již náležitě upraveny. Stále je ale vhodné si prohlédnout, jakým způsobem jsou označeny ve funkci *enum* kvůli označení používané ve funkcích již připravených k jejich implementaci.

4.3.1.2 Příprava dešifrovacího algoritmu

Implementovaci nově přidaného šifrovacího algoritmu je vhodné zanechat v externím souboru. Jeho podstatnou částí je funkce na dešifrování bloků, která obdrží parametrem v nějaké formě data k dešifrování, ukazatel na pole, kam uložit výsledek, hlavní a tweakovací klíč a číslo a velikost dešifrovaného sektoru. Je také nutné dodržovat správný operační mód šifry.

Přesnými specifikacemi samotné dešifrovací funkce se zde blíže zabývat nebudu, protože se pravděpodobně bude velmi lišit v závislosti na použité knihovně. Pro představu, jak funkce má pracovat, doporučuji sekci 4.2.2.2, kde je popsán průběh implementované šifry AES.

4.3.1.3 Soubor *tclib-crypto*

Soubor *tclib-crypto* obsahuje klíčovou funkci *tclib_decryptBlocks*, která volá konkrétní dešifrovací funkci v závislosti na vstupním parametru. Pro každou implementovanou šifru obsahuje jeden *case*, který pak volá konkrétni dešifrovací funkci. Kód pro jednotlivé šifry je závislý na jejím hesle v *enum* funkci zmíněné v 4.3.1.1.

První nutnou úpravou je *include* souboru obsahujícího implementaci přidáváné dešifrovací funkce do hlavičkového souboru *tclib-crypto.h*.

Druhou úpravou je pak přidání *case* pro přidávanou šifru. Název případu musí být shodný s řetězcem, který jsme zvolili pro označení nově implementované šifry v *tclib-includes.h*. V případě šifry Serpent a od ní odvozených kaskádových šifer stačí použít již připravený *case*. Obsahem *case* musí být

použití šifrovací funkce tak, aby na jejím konci byl výsledek (dešifrované pole bajtů) umístěno v ukazateli reprezentujícím výstupní pole bajtů (ukazatel *out* předaný jako parametr funkce *tclib_decryptBlocks*).

Pro implementaci kaskádových šifer je nutné pamatovat, že TrueCrypt používá velmi specifické pořadí klíčů a šifer v kaskádách. Zejména jde o princip, že pořadí šifer v názvu kaskády značí pořadí při dešifrování, tedy šifrování probíhá „obráceně“ a o vlastnost, že generovaný šifrovací klíč je rozdělený v půlce na hlavní a tweakovací část pro jednotlivé algoritmy, nejsou tedy pro jednotlivé šifry v blocích přímo za sebou. Bližší popis problematiky a příklad najdete v sekci 4.2.2.1.

4.3.2 Souborový systém

tc-lib v současné formě podporuje pouze souborové systémy FAT16 a FAT32. Při přidání dalších souborových systémů - například jedné z výše zmíněných NTFS či ext2/3/4 knihoven - je potřeba kromě zahrnutí nových externích knihoven upravit několik existujících funkcí a několik nových přidat.

4.3.2.1 Doplnění knihoven

Většina knihoven implementující souborové systémy vyžaduje doplnění funkcí pro čtení a zápis bloků specifické pro konkrétní zařízení. Při použití v knihovně tc-lib je nutné pamatovat, že čtený kontejner bude obsahovat zašifrované bloky dat. Ty je tedy nutno buď před samotným čtením nebo během něj dešifrovat pomocí šifrovacích klíčů získaných z hlavičkové části kontejneru. Proto je nutné čtecí funkci předávat strukturu *TCHeader* obsahující hlavičkové údaje. Druhou možností je dešifrovat bloky před čtením, v tom případě ale bude odlišná struktura *case* pro nově přidaný souborový systém v souboru *tclib-filesystem*.

D8le je potřeba mít na paměti, že samotný souborový systém začíná odsazený od začátku souboru. Je tedy nutné pro získání chtěného bloku odsadit velikost hlavičky v případě hlavního svazku a velikost hlavního svazku v případě skrytého. Toto odsazení je uloženo ve struktuře *TCHeader*.

4.3.2.2 Hlavičkový soubor *tclib-includes*

Hlavičkový soubor *tclib-includes.h*, ve kterém se nalézá definice většiny konstant a struktur využívaných knihobnou tc-lib, je prvním místem které je potřeba doplnit při implementaci nového souborového systému. Je zde potřeba doplnit funkci *enum* (enumerate, očíslovat) obsahující řetězce označující jednotlivé souborové systémy. Tento řetězec je pak v kódu nahrazován číslem, které mu funkce přidělí.

Druhým místem v tomto souboru je konstanta *MAX_FILENAME_SIZE*, která určuje maximální délku názvu jednoho souboru či složky libovolného souborového systému. Je používaná při definici struktury *file_info*. Pokud

je potřeba konstantu změnit, bude pravděpodobně nutné změnit i podobnou konstantu v již zahrnutých knihovnách, kde podobné omezení je nastavitelné.

Třetí podstatnou změnou je struktura *file_info*. Ta obsahuje veškeré dostupné informace o souborech jakéhokoliv souborového systému implementovaného knihovnou tc-lib. Pokud nově přidaný souborový systém podporuje jiné atributy souborů či složek, je nutné je do struktury přidat.

Soubor obsahuje i další konstanty, například velikost sektoru či nastavení typu pro uchovávání čísla sektoru. Tyto konstanty ale pravděpodobně budou pro většinu implementovaných souborových systémů stejné.

4.3.2.3 Příprava souborového systému

Kromě přidání základních operací pro čtení a zápis nutných pro samotnou funkci většiny knihoven implementujících souborové systémy je potřeba přidat ještě funkce specifické pro funkce knihovny tc-lib. Všechny tyto funkce jsou poté volány ze souboru *tclib-filesystem* při zadání tohoto konkrétního souborového systému.

První takovou funkcí je inicializační funkce daného souborového systému. Často je potřeba souborový systém nejdříve inicializovat než s ním lze provádět jakékoliv akce. Tyto inicializační funkce spolu s dalšími nutnými pro práci tc-lib (například uložení ukazatele na strukturu *TCHeader* nutnou k dešifrování) je vhodné umístit do samostatné funkce, která bude volána vždy při zahájení práce se souborovým systémem.

Druhou funkcí je uložení obsahu zadané složky daného souborového systému do pole struktur *file_info*. Toto pole si funkce sama naalokuje a poté na něj vrátí ukazatel. Kromě ukazatele vrátí také množství prvků v daném poli v parametru předanému funkci. Součástí tohoto je i přidání funkce, která informace získané externí knihovnou převede do struktury *file_info* používané knihovnou tc-lib.

Třetí funkcí je pak extrakce zvoleného souboru TrueCrypt kontejneru do externího souboru. Funkce se pokusí otevřít si k zápisu soubor na zadané cestě a pokud to jde, extrahuje do něj zadaný soubor zevnitř kontejneru. Návratovou hodnotu pak tvoří hodnota *true* pokud operace proběhla úspěšně či *false* pokud ne.

4.3.2.4 Soubor *tclib-filesystem*

Soubor *tclib-filesystem* funguje jako soubor spojující všechny použité souborové systémy a jejich funkce. Je tedy nutné zde přidat volání funkcí nově přidaného souborového systému, případně jejich základní úpravu do standartního tvaru.

Obsahuje funkce *init_fs* inicializující souborový systém, *get_directory* ukládající obsah zadané složky do pole struktur *file_info* a *extract_file* extrahující zadaná soubor z TrueCrypt kontejneru do zadaného externího souboru.

Všechny tyto funkce kromě jiných parametrů přijímají i typ (označení použité v *enum* funkci souboru *tclib-includes*) souborového systému na základě kterého pak volají funkce specifické pro daný souborový systém.

První nutnou úpravou je *include* souboru implementujícího nově přidávaný souborový systém do hlavičkového souboru *tclib-filesystem.h*

Pak je třeba přidat do všech tří funkcí *case* pro nově přidaný souborový systém. Je podstatné, aby dodržoval stejnou strukturu jako již implementované souborové systémy, zejména co se návratových hodnot týče. Funkce *init_fs* nevrací nic, zde by problém být neměl. *get_directory* musí vracet ukazatel na pole struktur *file_info* obsahující informace o jednotlivých prvcích zadané složky a jejich počet v parametru *item_count*. *extract_file* vrací hodnotu 1 při úspěchu, hodnotu 0 při chybě.

Je možné, že implementovaný souborový systém pracuje jinak s cestami, zejména jde o rozdíl přístupu pomocí kompletní cesty k souboru či nastavení pracovního adresáře a postupný přechod. Funkce souboru *tclib-filesystem* očekávají na vstupu kompletní cestu k žádané složce, pokud nově implementovaná knihovna neumí s kompletní cestou pracovat, je nutné tento problém vyřešit v rámci jí přiděleného *case*.

4.4 Nedostatky a další vývoj

Knihovna tc-lib má řadu nedostatků, ať už z důvodu nedostatku času na jejich doladění či chyb při jejím vytváření. Tyto nedostatky se zde pokusím stručně shrnout s krátkým vysvětlením, jak by mohla vypadat jejich oprava.

Prvním nedostatkem je chybějící podpora všech šifrovacích algoritmů použitých TrueCryptem. Patří zde zejména Serpent, který byl používán i v poslední verzi TrueCryptu, ale i starší již nepoužívané algoritmy, které v posledních verzích již nešlo použít k vytvoření nového kontejneru, ale stále šly pomocí TrueCryptu dešifrovat. Stejně tak chybí podpora dříve používaných operačních módů. Co je potřeba k implementaci nového šifrovacího algoritmu jsem stručně nastínil v kapitole 4.3.1.

Velmi podobně je na tom podpora souborových systémů, kdy v době psaní této práce knihovna tc-lib podporuje pouze souborové systémy FAT16 a FAT32. Co je potřeba udělat pro přidání dalšího souborového systému jsem popsal v kapitole 4.3.2.

V souvislosti se souborovými systémy má knihovna další velký nedostatek kterým je možnost otevřít v danou chvíli pouze jeden kontejner daného souborového systému. Tato nedokonalost vznikla v důsledku neznalosti přesných vlastností implementované knihovny pro souborový systém FAT16/32 Ultra-Embedded. Jednotlivé funkce sdružující souborové systémy tedy nedostávají jako parametr konkrétní instanci otevřeného kontejneru, s kterou mají pracovat, ale pouze jeho typ a počítá se s tím, že po inicializaci souborového systému bude už knihovna vědět, o který soubor jde. Pro opravu bude po-

třeba nejen změnit knihovnu Ultra-Embedded používanou pro implementaci FAT16/32, ale i strukturu funkcí v souboru *tclib_filesystem*. Dá se předpokládat, že každá nově implementovaná knihovna k práci se souborovými systémy bude používat odlišný způsob uchovávání konkrétní instance souborového systému. Pro předání instance bych tedy doporučil strukturu obsahující ukazatel na všechny typy instancí používaných souborových systémů spolu s přiznakem, která z nich je připravena k použití.

Knihovna není příliš optimalizovaná pro rychlost. Jde především o místa, kde je potřeba dešifrovat sektory datové části. Pro již implementované souborové systémy FAT16/32 si čtecí funkce neříká o dešifrování více sektorů zároveň, ale vždy si řekne pouze o jeden v cyklu přes počet sektorů. Při implementaci jsem měl obavy, zda by při čtení velkého množství dat najednou (například několik GB velkých souborů) program nenarazil na problémy s pamětí. Možnou změnou by byla úprava čtecí funkce *media_read* souboru *tclib-fat* tak, aby počet čtených bloků byl řízený konstantou.

V knihovně pak chybí podpora klíčových souborů (keyfiles). Jde o soubory, které lze předat TrueCryptu při vytváření nového svazku spolu s heslem a on pak obsah daných souborů použije při generování hlavičkového klíče. Při implementaci bude pravděpodobně potřeba jen přidat pomocnou funkci pro funkci *decrypt_header* souboru *tclib-header*, která použije zadané soubory a heslo ke generování pole bajtů, které pak bude používáno pro vytvoření hlavičkových klíčů.

Další vhodnou úpravou by bylo uložení cesty k souboru a typ souborového systému do struktury *TCHeader*. Důvodem je, že každá hlavička patří ke konkrétnímu TrueCrypt kontejneru a nedá se předpokládat, že by někdo použil údaje hlavičky jednoho souboru k dešifrování druhého. Poté by stačilo předávat funkcím souborových systémů pouze strukturu *TCHeader*, nebylo by potřeba předávat znovu cestu k souboru. Pro tuto úpravu by bylo potřeba změnit definici struktury *TCHeader* v souboru *tclib-includes* a funkce obsažené v souboru *tclib-filesystem* tak, aby využívaly údaje ze struktury místo předané parametry.

tc-lib podporuje pouze strukturu bajtů ve formě Little Endian. Ačkoliv to pravděpodobne nebude způsobovat problémy, protože TrueCrypt je podporovaný pouze systémy s Little Endian architekturou, mohlo by být vhodné pro úplnost knihovny přidat konstantu určující pořadí a podle toho se zařídit. Jde zejména o práci s klíči a dešifrovacími funkcemi.

4.5 Demonstrační program

Jako demonstrační program jsem se rozhodl implementovat alternativu dvou základních příkazů linuxové příkazové řádky - ls (list directory, vypiš obsah složky) a cp (copy, kopíruj). V knihovně se jejich implementace nachází v souborech *tclib-demo-ls* a *tclib-demo-cp*. Tyto příkazy jsem se rozhodl implemen-

tovat z toho důvodu, že dohromady názorně ukazují, jak s knihovnou pracovat a produkují i hmatatelný výstup.

Pro kompilaci demonstračního programu je připraven skript *compile*, který s parametrem *demo* zkompiluje výsledný program do souborů *tc-ls* a *tc-cp*

4.5.1 Popis funkce programu

Demonstrační program *tc-ls*, zkompilovaný s použitím knihovny tc-lib a implementovaný v souboru *tclib-demo-ls*, slouží k vypsání zadané složky TrueCrypt kontejneru. Program *tc-cp* implementovaný v souboru *tclib-demo-cp* extrahuje soubor ze zadaného kontejneru do externího souboru. Oba programy dohromady tvoří nástroj, který je sám o sobě vhodný i pro praktické účely - dovoluje najít žádaný soubor v kontejneru a extrahovat jej do zvoleného externího souboru. I přesto by šel v mnohém zdokonalit, například přidat interaktivní rozhraní pro uživatele.

Demonstrační program načítá heslo neskrytě přímo z příkazové řádky. Tento způsob zadávání není bezpečný, program slouží skutečně pouze pro názorné předvedení práce s knihovnou.

4.5.1.1 Výpis složky - *tc-ls*

Programu se předají jako povinné parametry po pořadě cesta ke kontejneru, heslo k jeho otevření a cesta, kterou chce uživatel vypsat. Dále je možné pomocí přepínačů ovlivnit formát výpisu. Přepínač -d (directory, složka) způsobí, že složky budou ve výpisu na začátku. Přepínač -s *kriterium* (sort, seřaď) seřadí výstup podle zadaného kritéria, podporované jsou n - název, s - velikost a m - doba poslední změny. Nakonec. přepínač -r (reverse, obráceně) obrátí pořadí výpisu složky. Výchozí nastavení je složky bez speciálního řazení složek, seřazení výstupu podle jména abecedně.

4.5.1.2 Extrakce souboru - *tc-cp*

Programu se předají jako povinné parametry po řadě cesta ke kontejneru, heslo k jeho otevření, cesta k souboru v kontejneru, který uživatel chce extrahovat a cesta v souborovém systémů, kam chce uživatel soubor uložit. Program nepodporuje žádné přepínače.

4.5.2 Implementace programu

Implementace využívá struktury *file_info* používané knihovnou tc-lib pro uchovávání údajů o obsahu složky. S pomocí těchto údajů pak řadí výstup dle volby uživatele. Velkou část souboru zabírají řadící funkce, které nejsou příliš zajímavé. Dále je zde definice struktury *sort_options*, která je využívaná k uchování předvoleb výpisu a následně použitá k řazení výstupu. Výpis je pak dělaný jednoduše jako cyklus přes všechny prvky pole struktur *file_info*.

Program nejprve zkontroluje uživatelský vstup z příkazové řádky. Pak se pokusí otevřít zadaný soubor ke čtení, získat z něj hlavičku, kterou mu funkce *decrypt_header* vrátí ve struktuře *TCHeader* a poté s její pomocí inicializovat souborový systém. Následně si program pouze řekne pomocí funkce *get_directory* o uložení obsahu složky do předem připraveného ukazatele na pole struktur *file_info*. Toto pole je následně seřazeno a vypsáno pomocí pomocných funkcí programu *tc-ls*.

Extrakce souboru je velmi podobná výpisu složky, jen se místo zavolání funkce pro uložení obsahu složky zavolá funkce *extract_file* pro extrakci souboru z kontejneru s parametry přímo předanými uživatelem.

Závěr

Výsledkem této bakalářské práce je funkční knihovna podporující off-line práci s TrueCrypt kontejnery. Je implementována v programovacím jazyku C a určena pro operační systém Linux. Podporuje šifrovací algoritmy AES, Twofish a AES-Twofish kaskádovou šifru a souborové systémy FAT16 a FAT32. Knihovna je strukturována tak, aby rozšíření o další šifrovací algoritmus či souborový systém vyžadovalo pouze malé změny v původním kódu. Součástí knihovny je také demonstrační program umožňující výpis zvolené složky a extrakci zvoleného souboru TrueCrypt kontejneru. Cíl práce byl tímto splněn.

Knihovna má jisté nedostatky, které by bylo vhodné změnit nebo řešit jinak. Některé z TrueCryptem podporovaných funkcí také v knihovně úplně chybí. Tyto problémy vznikly zejména kvůli nedokonalém návrhu, nepřesné znalosti jednotlivých knihoven či nedostatku času pro implementaci všech částí původního TrueCryptu.

Největším problémem při dalším rozšíření knihovny bude nejspíše nedostatek vhodných knihoven pro implementaci dalších souborových systémů. Řada programů nabízejících práci s nimi je totiž již kompletním ovladačem specifickým pro určité zařízení, což odporuje požadavku na off-line zpracování kontejneru. Je také možné, že půjde o vhodnou knihovnu, která ale bude vyžadovat větší úpravy kvůli zahrnutí dešifrovacích algoritmů do čtěcích funkcí.

Vývoj aplikace pro mě byl přínosem. Při práci jsem se dozvěděl nejen jak přesně fungují samotné šifrovací aplikace, ale i podrobnější informace o funkci jednotlivých šifrovacích algoritmů, operačních módů či souborových systémů. Další novou zkušeností bylo vytváření delšího kódu od prvotního návrhu struktury a hledání vhodných knihoven až po konečnou implementaci. Knihovnu bych rád ve svém volném čase rozšířil o další souborové systémy a šifrovací algoritmy tak, aby šlo o hotový celek vhodný k použití.

Literatura

[1] Wikimedia Commons: AES Subbytes, ShiftRows, MixColumns, AddRoundKey [online]. January 2005, [Cit. 2015-05-01]. Dostupné z: http://commons.wikimedia.org/wiki/Category:AES

[2] Phipps, S.: TrueCrypt or false? Would-be open source project must clean up its act [online]. November 2013, [Cit. 2015-03-02]. Dostupné z: http://www.infoworld.com/article/2609745/open-source-software/truecrypt-or-false--would-be-open-source-project-must-clean-up-its-act.html

[3] TrueCrypt Team: P. Le Roux (author of E4M) accused by W.Hafner (SecurStar) [online]. February 2004, [Cit. 2015-03-02]. Dostupné z: https://groups.google.com/d/msg/alt.security.scramdisk/HYa8Wb_4acs/GEyw8fJi9csJ

[4] TrueCrypt Foundation: Version History - TrueCrypt Documentation [online]. February 2015, [Cit. 2015-03-02]. Dostupné z: http://andryou.com/truecrypt/docs/version-history2.php

[5] Palazzolo, J.: Court: Fifth Amendment Protects Suspects from Having to Decrypt Hard Drives [online]. February 2012, [Cit. 2015-03-07]. Dostupné z: http://blogs.wsj.com/law/2012/02/23/court-fifth-amendment-protects-suspects-from-decrypting-computers/

[6] Junestam, A.; Guigo, N.: Open Crypto Audit Project TrueCrypt Security Assessment [online]. February 2014, [Cit. 2015-03-07]. Dostupné z: https://opencryptoaudit.org/reports/iSec_Final_Open_Crypto_Audit_Project_TrueCrypt_Security_Assessment.pdf

[7] Balducci, A.; Devlin, S.; Ritter, T.: Open Crypto Audit Project TrueCrypt Cryptographic Review [online]. March 2015, [Cit. 2015-05-01]. Dostupné z: https://opencryptoaudit.org/reports/TrueCrypt_Phase_II_NCC_OCAP_final.pdf

[8] Green, M.: [Here is the note I sent...] In: Twitter [online]. June 2014, [Cit. 2015-03-07], odkaz v tweetu. Dostupné z: https://twitter.com/matthew_d_green/status/478721271316758528

[9] TrueCrypt Foundation: TrueCrypt - Volume Format Specification [online]. October 2013, [Cit. 2015-03-11]. Dostupné z: http://andryou.com/truecrypt_orig/docs/volume-format-specification/

[10] TrueCrypt Foundation: [Program Menu - TrueCrypt Documentation] Tools -> Restore Volume Header [online]. February 2015, [Cit. 2015-03-11]. Dostupné z: http://andryou.com/truecrypt/docs/program-menu.php

[11] TrueCrypt Foundation: Header Key Derivation - TrueCrypt Documentation [online]. February 2015, [Cit. 2015-04-07]. Dostupné z: http://andryou.com/truecrypt/docs/header-key-derivation.php

[12] Kaliski, B.: PKCS number 5: Password-Based Cryptography Specification, Version 2.0 [online]. September 2000, [Cit. 2015-04-07]. Dostupné z: http://www.rfc-base.org/txt/rfc-2898.txt

[13] Krawczyk, H.; Bellare, M.; Canetti, R.: HMAC: Keyed-Hashing for Message Authentication [online]. Februrary 1997, [Cit. 2015-04-07]. Dostupné z: http://www.rfc-base.org/txt/rfc-2104.txt

[14] Paar, C.; Pelzl, J.: *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010.

[15] Penard, W.; van Werkhoven, T.: On the Secure Hash Algorithm family [online]. January 2008, [Cit. 2015-04-07]. Dostupné z: http://www.staff.science.uu.nl/~werkh108/docs/study/Y5_07_08/infocry/project/Cryp08.pdf

[16] Preneel, B.; Dobbertin, H.; Bosselaers, A.: The Cryptographic Hash Function - RIPEMD-160 [online]. September 1997, [Cit. 2015-04-07]. Dostupné z: https://www.cosic.esat.kuleuven.be/publications/article-317.pdf

[17] Bosselaers, A.: The RIPEMD-160 page [online]. February 2012, [Cit. 2015-04-07]. Dostupné z: http://homes.esat.kuleuven.be/~bosselae/ripemd160.html

[18] Barreto, P. S.; Rijmen, V.: The WHIRLPOOL Hashing Function [online]. May 2003, [Cit. 2015-04-12]. Dostupné z: http://www.larc.usp.br/~pbarreto/whirlpool.zip

[19] Albertson, D.; Babinkostova, L.; de Kergorlay, H.; aj.: Generalizing the WHIRLPOOL Hash Function [online]. July 2013, [Cit. 2015-04-12]. Dostupné z: http://math.boisestate.edu/reu/publications/WhirlPool.pdf

[20] TrueCrypt Foundation: Encryption Algorithms - TrueCrypt Documentation [online]. February 2015, [Cit. 2015-04-07]. Dostupné z: http://andryou.com/truecrypt/docs/encryption-algorithms.php

[21] Menezes, A.; van Oorschot =and S. Vanstone, P.: Handbook of Applied Cryptography, kapitola 7.2.2 [online]. October 1996, [Cit. 2015-04-12]. Dostupné z: http://cacr.uwaterloo.ca/hac/about/chap7.pdf#page=7

[22] Dworkin, M.: Recommendation for Block Cipher Modes of Operation - Methods and Techniques, kapitola 6.2 [online]. December 2001, [Cit. 2015-04-12]. Dostupné z: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf#page=17

[23] Liskov, M.; Rivest, R. L.; Wagner, D.: Tweakable Block Ciphers [online]. January 2002, [Cit. 2015-04-12]. Dostupné z: https://www.eecs.berkeley.edu/~daw/papers/tweak-crypto02.pdf

[24] NIST: Proposal To Extend CBC Mode By Ciphertext Stealing [online]. May 2007, [Cit. 2015-04-12]. Dostupné z: http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/ciphertext%20stealing%20proposal.pdf

[25] Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC [online]. September 2004, [Cit. 2015-04-12]. Dostupné z: http://web.cs.ucdavis.edu/~rogaway/papers/offsets.pdf

[26] Trenholme, S.: Rijndael's key schedule [online]. 2005, [Cit. 2015-04-15]. Dostupné z: http://www.samiam.org/key-schedule.html

[27] NIST: Announcing the ADVANCED ENCRYPTION STANDARD (AES) [online]. November 2001, [Cit. 2015-04-15]. Dostupné z: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[28] Anderson, R.; Biham, E.; Knudsen, L.: Serpent: A Proposal for the Advanced Encryption Standard [online]. July 2006, [Cit. 2015-04-20]. Dostupné z: http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf

[29] Moore, T.; Ballentine, K.: Serpent Cipher Design and Analysis [online]. November 2012, [Cit. 2015-04-20]. Dostupné z: http://www.cs.rit.edu/~kxb3695/csc482/report.pdf

[30] Schneier, B.; Kelsey, J.; Whiting, D.; aj.: Twofish: A 128-Bit Block Cipher [online]. June 1998, [Cit. 2015-04-20]. Dostupné z: https://www.schneier.com/paper-twofish-paper.pdf

[31] Dancraggs: Twofish diagram [online]. January 2008, [Cit. 2015-04-20]. Dostupné z: http://upload.wikimedia.org/wikipedia/commons/e/ee/Twofishalgo.svg

[32] Microsoft Corporation: FAT32 File System [online]. 2005, [Cit. 2015-04-22]. Dostupné z: http://web.archive.org/web/20050319235548/www.microsoft.com/resources/documentation/Windows/XP/all/reskit/en-us/prkc_fil_cycz.asp

[33] ChaN: FatFs - Generic FAT File System Module [online]. February 2015, [Cit. 2015-04-22]. Dostupné z: http://elm-chan.org/fsw/ff/00index_e.html

[34] Ultra-Embedded: FAT16/32 File System Library [online]. November 2013, [Cit. 2015-04-22]. Dostupné z: http://ultra-embedded.com/fat_filelib

[35] Micro Digital Inc.: smxFS^{TM}- Portable FAT File System [online]. February 2011, [Cit. 2015-04-22]. Dostupné z: http://www.smxrtos.com/rtos/fileio/smxfs.pdf

[36] On Time: RTFiles-32 - Portable FAT, exFAT, and ISO 9660 File System for Embedded Systems [online]. April 2015, [Cit. 2015-04-22]. Dostupné z: http://www.on-time.com/rtfiles-32.htm

[37] LSoft Technologies Inc.: NTFS - New Technology File System [online]. April 2015, [Cit. 2015-04-24]. Dostupné z: http://ntfs.com/ntfs.htm

[38] Tuxera Inc.: Open Source: NTFS-3G - Read-Write NTFS Driver [online]. March 2015, [Cit. 2015-04-24]. Dostupné z: http://www.tuxera.com/community/open-source-ntfs-3g/

[39] Paragon Technologie GmbH.: NTFS for Linux 7.0 Embedded Solution [online]. March 2015, [Cit. 2015-04-24]. Dostupné z: https://www.paragon-software.com/technologies/components/ntfs-linux-embedded/

[40] Wu, M.: Ext2Fsd Project [online]. August 2014, [Cit. 2015-04-25]. Dostupné z: http://www.ext2fsd.com/

[41] Paragon Technologie GmbH.: ExtFS for Windows ®Professional [online]. March 2015, [Cit. 2015-04-24]. Dostupné z: http://www.paragon-software.com/home/extfs-windows-pro/

[42] Paragon Technologie GmbH.: ExtFS for Mac OS X 9 [online]. March 2015, [Cit. 2015-04-24]. Dostupné z: http://www.paragon-software.com/home/extfs-mac/

[43] TrueCrypt Foundation: Cascades - TrueCrypt Documentation [online]. February 2015, [Cit. 2015-04-28]. Dostupné z: http://andryou.com/truecrypt/docs/cascades.php

Příloha **A**

Obsah přiloženého CD

- `README_unix.txt` ..... popis obsahu CD a knihovny v Unix formátu
- `README_win.txt` ..... popis obsahu CD a knihovny ve Windows formátu
- `ZZP.pdf` ..... zadání práce
- `BP.pdf` ..... tento dokument
- `doc` ..... adresář se soubory potřebnými pro kompilaci tohoto dokumentu
- `src`
  - `fat` ..... knihovna Ultra-Embedded pro práci s FAT
  - `testfiles` ..... několik kontejnerů pro zkoušku programu
  - `compile` ..... skript pro kompilaci
  - `tc-ls` ..... demonstrační program – výpis složky
  - `tc-cp` ..... demonstrační program – extrakce souboru
  - `*.c` ..... implementace knihovny tclib