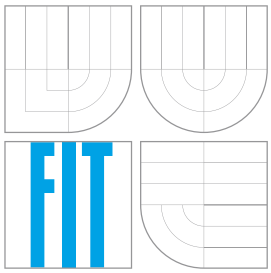
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ANALÝZA ŠIFROVACÍCH METOD PRO KLONOVÁNÍ DISKŮ

ANALYSIS OF ENCIPHEREMENT METHODS FOR DISK CLONING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JIŘÍ CHROMEČKA
AUTHOR

VEDOUCÍ PRÁCE doc. Ing., Dipl.-Ing. MARTIN DRAHANSKÝ, Ph.D.
SUPERVISOR

BRNO 2016

Zadání diplomové práce/17976/2015/xchrom12

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Chromečka Jiří, Bc.**
Obor: Bezpečnost informačních technologií
Téma: **Analýza šifrovacích metod pro klonování disků**
Analysis of Encipherement Methods for Disk Cloning
Kategorie: Bezpečnost

Pokyny:

1. Prostudujte literaturu týkající se šifrování dat (především algoritmem AES). Zaměřte se zejména na šifrování disků, adresářů a souborů. Zjistěte možnosti skriptování šifrování pro účely klonování disků.
2. Navrhněte aplikační řešení pro šifrování disků a adresářů za účelem klonování disků pro platformu Windows.
3. Vámi navržené řešení z předchozího bodu implementujte v C/C++/C#. K implementaci využijte volně dostupné zdrojové kódy pro algoritmus AES, které začleňte do Vaší aplikace. Aplikace musí podporovat vícejazyčnost. Klíčem pro šifrování bude vstupní heslo, u kterého ověřte jeho sílu (bezpečnost).
4. Vaše řešení otestujte na 4 až 5 noteboocích s Win 8.1 a Win 10. Zaměřte se na funkčnost a rychlost (v závislosti na množství dat).
5. Shrňte a diskutujte dosažené výsledky.

Literatura:

- Shaska T.: *Introduction to Cryptography*. 2005, s. 115.
- Zeghid M., Machhout M., Khriji L., Baganne A., Tourki R.: *A Modified AES Based Algorithm for Image Encryption*. International Journal of Computer Science and Engineering, Vol. 1, No. 1, 2007, s. 70-75, ISSN 0976-2760.
- Macek R. *Bezpečnostní aspekty souborových systémů a bezpečnost dat zajištěná šifrováním disků*. BP, FI MU, 2011, s. 43.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Drahanský Martin, doc. Ing., Dipl.-Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Předložený text se zabývá návrhem vícejazyčné aplikace a její následnou implementací v jazyce C++. Aplikace šifruje diskové oddíly se systémem Microsoft Windows užitím symetrické kryptografie a autentizace pomocí hesla, jehož síla zabezpečení je ověřena. Zabezpečení citlivých dat je v oblasti informačních technologií jeden z důležitých bezpečnostních cílů. Symetrická kryptografie, která používá stejný klíč pro šifrování i dešifrování, je díky své rychlosti vhodná pro šifrování datového úložiště. Pro zvýšení bezpečnosti je možné šifrovat celý diskový oddíl s citlivými daty.

Abstract

The presented text deals with designing of a multilingual application and its following implementation in the C++ language. The application encrypts disk volumes with Microsoft Windows system using symmetric cryptography and password authentication, where the password security strength is verified. Securing the sensitive data is one of the important security goals in area of information technology. The symmetric cryptography uses the same key for both the encryption and the decryption and due to its speed it is suitable for the data storage encryption. For the higher security it is possible to encrypt a whole disk volume with sensitive data.

Klíčová slova

šifrování disku, symetrická kryptografie, AES, 3DES, Skipjack, C++, Crypto++, Windows, Fujitsu

Keywords

disk encryption, symmetric cryptography, AES, 3DES, Skipjack, C++, Crypto++, Windows, Fujitsu

Citace

CHROMEČKA, Jiří. *Analýza šifrovacích metod pro klonování disků*. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Drahanský Martin.

Analýza šifrovacích metod pro klonování disků

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana docenta Martina Drahanského. Další informace mi poskytl zadavatel pan inženýr Martin Černý. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Chromečka
21. května 2016

Poděkování

Děkuji především svému vedoucímu panu docentovi Martinovi Drahanskému za jeho vstřícnost, zprostředkování komunikace s firmou Fujitsu a jeho cenné rady, panu inženýru Martinovi Černému za konzultace a téma práce, svému známému Tomáši Růžičkovi za pomoc při programování a také své rodině za podporu, kterou mi poskytli.

Obsah

Kapitola 1

Úvod

V dnešní době si ukládáme informace různé důležitosti do elektronické formy v podobě dat. Některá data jsou neškodná a mohou být volně dostupná, jiná data jsou soukromá a nechceme, aby se dostala do rukou nepovolaným osobám. Z tohoto důvodu se v oblasti bezpečnosti informačních technologií zabýváme ochranou a způsoby zabezpečení těchto dat. Neustále se vyvíjejí nové a vylepšené metody, protože i druhá strana přichází se slabými místy, která je možné zneužít. Výstupem této práce je aplikace, která zajistí, že citlivá data se nebudou za žádných okolností nacházet nechráněná na nezabezpečeném datovém úložišti.

Jedním ze způsobů, jak tohoto cíle dosáhnout, je udržovat data zašifrovaná, aby kdokoli, kdo k nim chce získat přístup, musel znát klíč k jejich dešifrování. Moderní kryptografie rozlišuje dva typy šifer [5]: symetrické a asymetrické. U symetrických šifer [5] je možné použít stejný klíč pro zašifrování i dešifrování dat, zatímco u asymetrických šifer [5] existují dva různé klíče pro každou úlohu. Symetrické šifry dále rozdělujeme na proudové a blokové, podle toho, zda šifrujeme data postupně, anebo po blocích stejné velikosti. Tato práce se zabývá právě symetrickými blokovými šiframi, které podrobněji popisuje **Kapitola** 2.1.

Dále budeme předpokládat, že datová úložiště jsou pevné disky s operačním systémem Windows, neboť většina aplikací využívajících šifrovací algoritmy je závislá na zvolené platformě. Data na disku je možné ponechat zašifrovaná v určité oblasti jako je například adresář, anebo lze zašifrovat celý oddíl s daty (i systémový) a tato práce požaduje právě tuto druhou možnost, kvůli zvýšené bezpečnosti za cenu vyšší časové náročnosti. Více se tomuto tématu věnuje **Kapitola** 2.2.

V současnosti se můžeme setkat s různými nástroji, které nám umožňují zabezpečit různá datová úložiště a **Kapitola** 3 zmiňuje některé z nich, zaměřené na námi zvolenou platformu a porovnává jejich výhody a nevýhody.

O návrhu této aplikace, ve které využijeme znalosti uvedené v úvodních teoretických kapitolách, pojednává **Kapitola** 4. Tím se dostáváme k jádru samotné práce. Zde se specifikuje rozhraní, pomocí kterého spustíme proces šifrování, vybrané metody a šifrovací algoritmy.

Praktickou implementací v jazyce C++ se následně zabývá **Kapitola** 5, ve které rozebíráme strukturu kódu. Využívá se rozsáhlé a volně dostupné knihovny `Crypto++`, která implementuje různé šifrovací algoritmy a poskytuje vlastní rozhraní, které lze začlenit do naší aplikace.

Výsledky testování této aplikace podává a diskutuje **Kapitola** 6. Protože se tato aplikace využije v masovém nasazení, je její univerzálnost v rámci dané platformy důležitá, a proto testujeme i různé verze operačního systému Windows.

Závěrečné zhodnocení dosažených cílů a nástin možných rozšíření shrnuje **Kapitola** 7.

Kapitola 2

Šifrování

Základní úlohou kryptografie [20] je poskytovat *důvěrnost* prostřednictvím šifrovacích algoritmů. Rozlišujeme šifrování, jehož vstupem je nechráněná zpráva zvaná *plaintext*, obvykle také šifrovací klíč a výstupem je zabezpečená zpráva zvaná *ciphertext*, a dešifrování, které provádí inverzní operaci k šifrování.

Tato oblast je velmi stará, příkladem může být slavná *Césarova šifra* [5], která využívá posuv všech písmen zprávy o konstantní počet míst v abecedě. Moderní kryptografie, kterou se budeme dále zabývat, se obvykle dělí na [5, 20]:

- *Symetrickou kryptografii (Symmetric-Key Cryptography)*,
- *kryptografii s veřejným klíčem (Public-Key Cryptography)*.

Symetrická kryptografie [5] využívá podle svého jména stejný *tajný klíč* pro šifrování i dešifrování. Je méně výpočetně náročná a efektivnější, než kryptografie s veřejným klíčem, a proto je vhodná pro šifrování velkého množství dat, nicméně vyžaduje znalost šifrovacího klíče na obou stranách, takže je nutné se nejprve bezpečným způsobem domluvit na klíči. Kryptografie s veřejným klíčem, používá dva klíče - *soukromý* a *veřejný*. Jeden se používá na šifrování a druhý na dešifrování, podle situace. Obvykle se tyto postupy navíc kombinují.

Rozeznáváme celkem čtyři bezpečnostní cíle, které jsou obzvláště důležité u asymetrické kryptografie [20]:

- *Důvěrnost (confidentiality)* zaručuje, že nikdo nepovolaný se nedozví obsah zprávy,
- *integrita dat (data integrity)* zamezuje, aby byla zpráva v průběhu přenosu změněna,
- *autentizace (authentication)* podává informaci o původci zprávy,
- *nepopiratelnost (non-repudiation)* zajišťuje, že původce opravdu odeslal tuto zprávu.

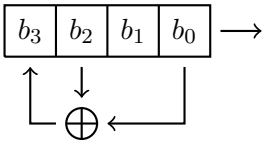
Útoky na kryptografická schémata se zabývá kryptoanalýza [5], kterou společně s kryptografií označujeme za kryptologii. Velmi důležitý předpoklad zvaný *Kerckhoffův Princip* [5] říká, že útočník zná všechny podrobnosti o kryptografickém systému, včetně algoritmů a jejich implementace, a tedy jeho bezpečnost musí záviset pouze na tajemství šifrovacích klíčů.

2.1 Symetrická kryptografie

U symetrické kryptografie je zapotřebí jediný klíč pro šifrování i dešifrování, a tedy pomocí daného klíče vždy dostaneme unikátní výstup. Rozlišujeme dva druhy šifer [5]:

- *Proudové šifry (Stream Ciphers)*,
- *blokové šifry (Block Ciphers)*.

U proudové šifer [5] je každý znak otevřené zprávy šifrován pomocí odpovídajícího znaku z toku pseudonáhodných čísel zvaného *keystream*. Často jsou proudové šifry založené na *LFSR* (Linear Feedback Shift Register, posuvný registr s lineární zpětnou vazbou) s prostou operací $\oplus$, viz dále.



Obrázek 2.1: Příklad čtyřbitového LFSR.

Za dokonalou nerozluštitelnou šifru je považována *Vernamova šifra (One-time pad)* [5], poprvé popsána Frankem Millerem [2], která musí splňovat následující tři pravidla:

1. keystream je **stejně dlouhý** jako zpráva,
2. keystream je **opravdu náhodný**,
3. keystream se **nikdy neopakuje**.

Blokové šifry [5] na rozdíl od proudových šifrují po blocích dat konstantní délky. Mějme blok otevřené zprávy M a blok zašifrované zprávy C, obsah $M = C = \{0,1\}^n$ a klíč $K = \{0,1\}^r$. Potom šifrování E a dešifrování D definujeme jako následující zobrazení:

$$E\colon \quad K \times M \to C \quad = \quad \{0,1\}^r \times \{0,1\}^n \to \{0,1\}^n \tag{2.1}$$

$$D\colon \quad K \times C \to M \quad = \quad \{0,1\}^r \times \{0,1\}^n \to \{0,1\}^n \tag{2.2}$$

Zde se objevuje jeden z nedostatků blokových šifer [5], protože existuje 2^n bloků otevřených i zašifrovaných zpráv, mezi kterými je bijektivní zobrazení, a tedy mluvíme o permutaci. Nicméně počet permutací je výrazně omezen volbou klíče z 2^r možných, ale klíč musí mít r bitů, což ve výsledku tvoří velmi malou množinu klíčů a nelze mít dokonalou blokovou šifru. Stále lze zvýšit bezpečnost, když použijeme generování opravdu náhodných čísel.

Velmi častá binární operace v oblasti šifrování je bitový exkluzivní součet, který značíme symbolem $\oplus$. Sémantika této operace je uvedena v Tabulce 2.1.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

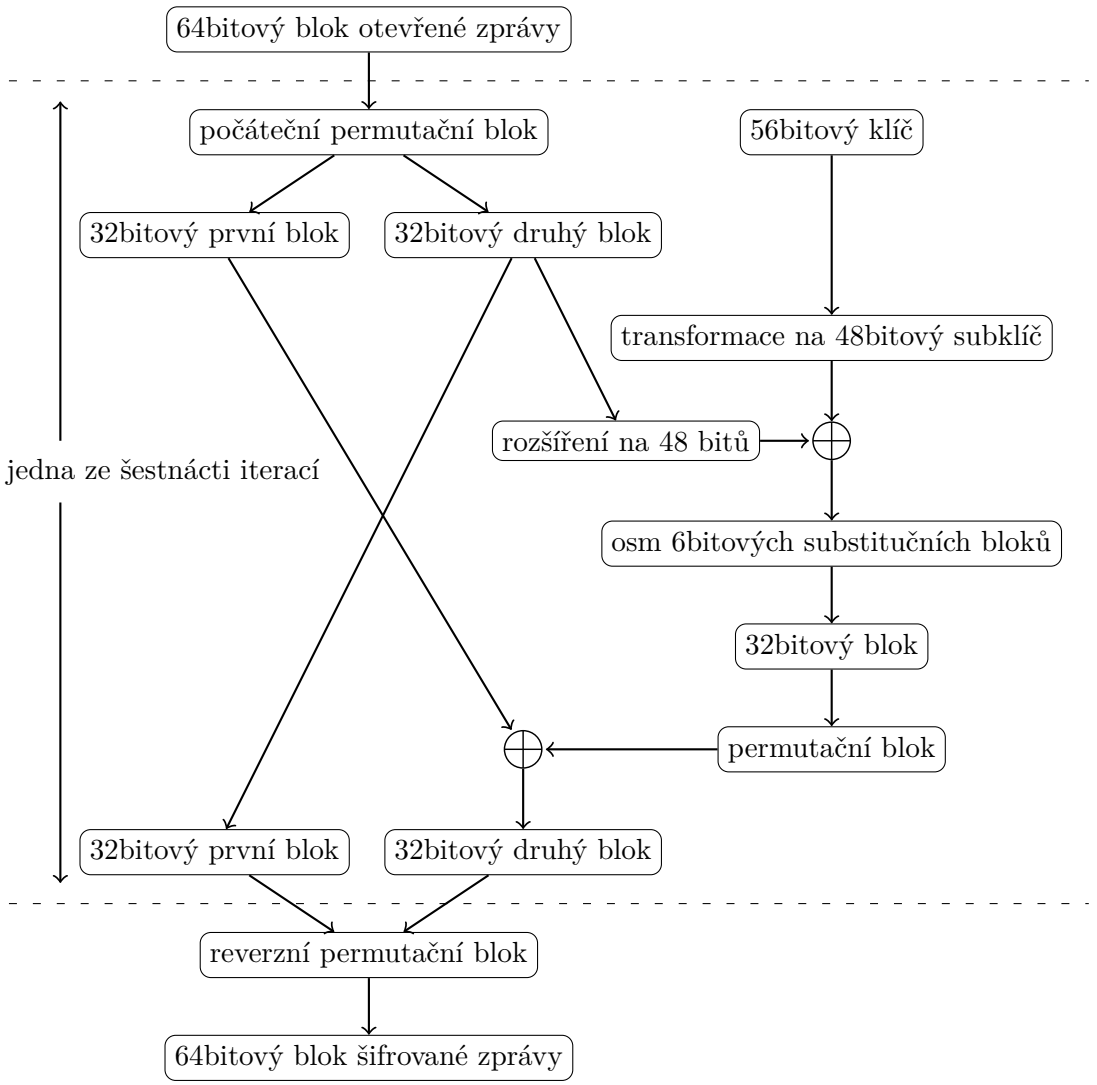
Tabulka 2.1: Sémantika bitového exkluzivního součtu $\oplus$.

V současnosti jsou organizací NIST schválené tři blokové šifry [27]: *3DES*, *AES* a *Skipjack*, které budou popsány v následujícím textu.

2.1.1 DES

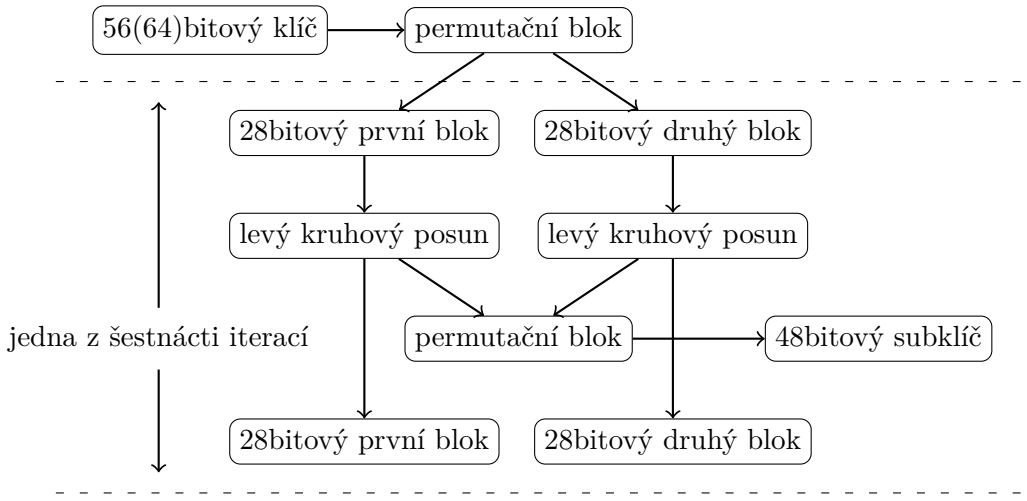
Data Encryption Standard byl poprvé standardizován v roce 1977 (FIPS 46) a až do roku 2005 oficiálně schválen organizací NIST (poslední specifikace je z roku 1999 [26]). Délka klíče je 56 bitů a bloku otevřené zprávy 64 bitů. Průběh šifrování sestává ze šestnácti iterací a pro každou z nich se vypočítá 48bitový subklíč, pomocí kterého se šifruje 32bitový blok zprávy.

Na začátku šifrování je vstupní 64bitový blok zprávy permutován a rozdělen na dvě 32bitové poloviny. 32bitový první blok je nejdříve rozšířen na 48 bitů. Poté se nad blokem a subklíčem volá $\oplus$, výsledných 48 bitů je rozděleno do osmi skupin po šesti bitech a každá skupina je daným způsobem substituována čtyřmi bity. Tím dostaneme 32 bitů, které jsou následně znovu permutovány. Výsledek této permutace opět pomocí funkce $\oplus$ s druhým vstupním blokem bude tvořit druhý blok pro další iteraci. První blok nové iterace tvoří druhý blok z předchozí iterace. Po šestnácti iteracích pomocí reverzního permutačního bloku vzhledem k počátečnímu dostaneme výsledný blok šifrované zprávy. Postup operace znázorňuje Obrázek 2.2.



Obrázek 2.2: Schéma šifrování pomocí DES.

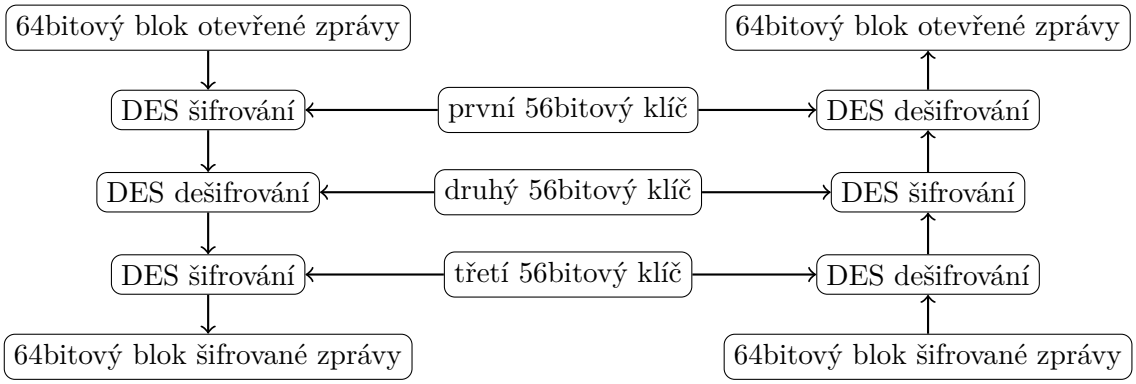
Transformace 56bitového klíče na 48bitový subklíč využívá posuvy a permutace. Protože klíč bývá zarovnán na 64 bitů, které tak zahrnují i paritní bity, počáteční operací je permutace, která zvolí 56 bitů a rozdělí je na dvě 25bitové poloviny. Nad těmito dvěma polovinami jsou provedeny dva různé kruhové posuvy doleva. Výsledné dvě poloviny jsou poté spojeny a pomocí další permutace tvoří 48bitový subklíč pro danou iteraci. Obě poloviny jsou zároveň vstupem do další iterace. Postup ukazuje Obrázek 2.3.



Obrázek 2.3: Generování subklíčů DES.

Dešifrování bloku zprávy se od jeho šifrování liší pouze opačným pořadím 48bitových subklíčů, než ve kterém pořadí byly tyto subklíče použity při šifrování bloku zprávy. Pro DES byly organizací NIST oficiálně specifikovány [22] použití v módech ECB, CMC, CFB a OFB, které budou uvedeny v Kapitole 2.1.4.

Pro zvýšení odolnosti vůči prolomení byla v roce 1999 schválena organizací NIST varianta zvaná `3DES` (poslední specifikace je z roku 2012 [33]), která místo jednoho klíče používá celkem tři 56bitové klíče, které se většinou liší. Pokud by všechny klíče byly stejné, tato varianta by se od původní DES nijak nelišila. Při šifrování bloku zprávy nejprve použijeme první klíč, výsledek dešifrujeme druhým klíčem, a poté znovu šifrujeme třetím klíčem. Postup dešifrování je symetrický, nejdříve dešifrujeme třetím klíčem, poté šifrujeme druhým klíčem a nakonec dešifrujeme prvním klíčem. Schéma znázorňuje Obrázek 2.4.



Obrázek 2.4: Schéma šifrování (vlevo) a dešifrování (vpravo) pomocí 3DES.

2.1.2 AES

Advanced Encryption Standard [24] byl organizací NIST v roce 2000 stanoven nástupcem DES. Do výběrového procesu [24] byly zařazeny různé blokové šifry, mj. *Twofish*, *Serpent* a *Rijndael*, z nichž právě šifrovací algoritmus *Rijndael* [4, 28], jehož autory jsou Joan Daemen a Vincent Rijmen, byl zvolen jako AES. *Rijndael* podporuje různé kombinace velikostí bloků a klíčů od 128 bitů do 256 bitů. AES toto omezil na velikost bloku 128 bitů a tři velikosti klíčů - 128bitový, 192bitový a 256bitový.

Podobně jako u DES je AES iterativní bloková šifra s 10/12/14 iteracemi v závislosti na velikosti klíče 128/192/256 bitů. AES v každé iteraci pracuje se 128bitovým blokem jako s maticí velikosti 4×4 s buňkami o velikosti 8 bitů.

V průběhu jednotlivých iterací opět využíváme operaci $\oplus$ buněk matice se subklíčem (viz Tabulka 2.1). Další operace pojmenujeme následovně:

- `MixColumns`

  Invertibilní multiplikativní transformace matice se sloupci jako polynomy.

- `ShiftRows`

  Kruhový posun řádků matice doleva o různou hodnotu 0/1/2/3 podle čísla řádku. Operace je invertibilní, postačí kruhový posun doprava o stejnou hodnotu.

- `SubBytes`

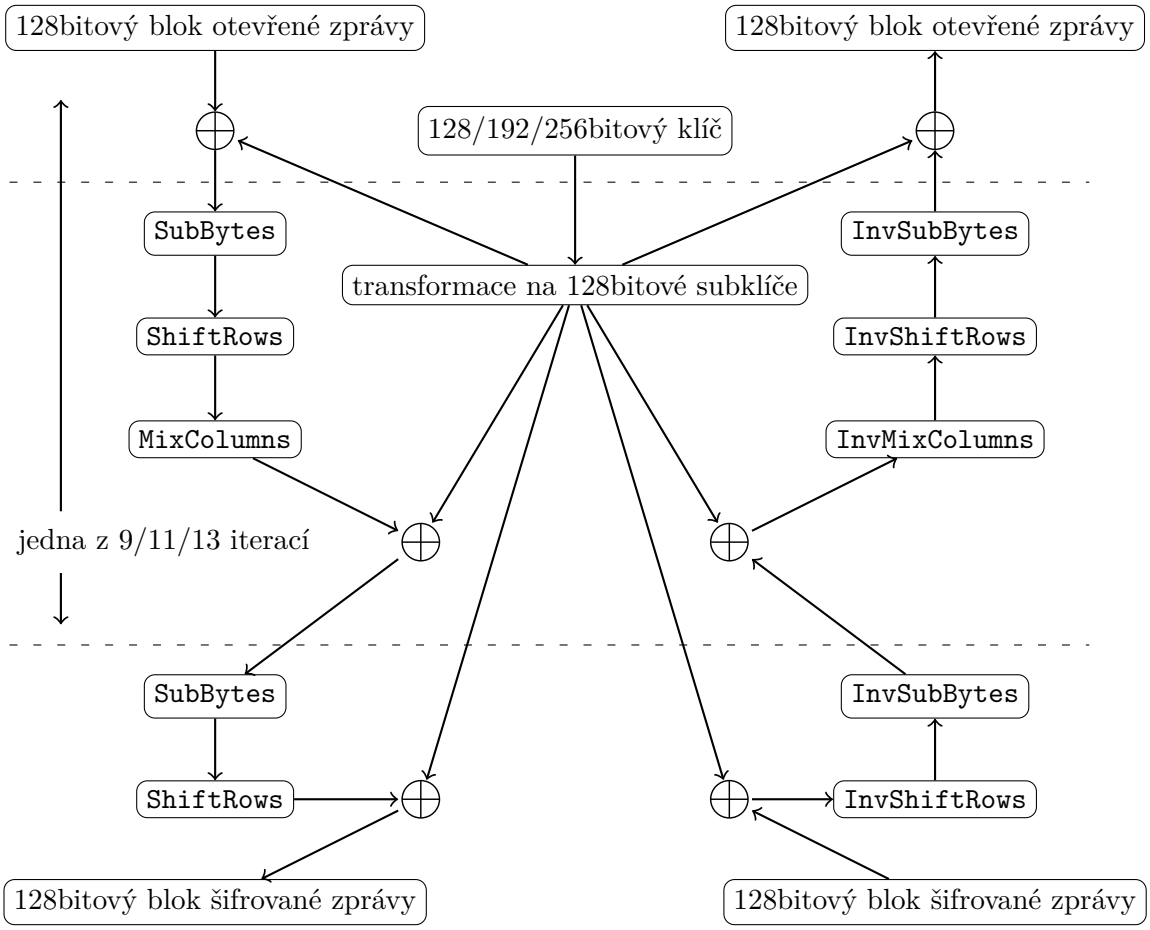
  Substituce každé buňky matice pomocí multiplikativní inverze a afinní transformace.

Při šifrování 128bitového bloku otevřené zprávy nejprve převedeme blok na matici a inicializujeme ji pomocí operace $\oplus$ se subklíčem, jehož generování bude zmíněno dále. Poté v každé iteraci mimo té poslední provedeme v tomto pořadí následující operace - `SubBytes`, `ShiftRows`, `MixColumns` a $\oplus$ matice s novým subklíčem. Poslední iterace se od ostatních mírně liší, protože je vynechána operace `MixColumns`.

Při dešifrování 128bitového bloku šifrované zprávy provádíme inverzní operace (budeme značit prefixem `Inv`) v opačném pořadí vzhledem k jejímu šifrování. V první iteraci nejprve provedeme operace $\oplus$ matice se subklíčem, `InvShiftRows` a `InvSubBytes`. Ve zbylých iteracích provedeme operace $\oplus$ matice se subklíčem, `InvMixColumns`, `InvShiftRows` a `InvSubBytes`. Nakonec provedeme poslední operaci $\oplus$ matice se subklíčem. Toto schéma šifrování a dešifrování pomocí AES lze vidět na Obrázku 2.5.

Transformaci klíče na 128bitové subklíče pro jednotlivé iterace obvykle označujeme za jeho expanzi, kdy vzniklé subklíče uspořádáváme za sebou, takže tvoří jeden dlouhý řetězec. V rámci algoritmu potřebujeme inicializační subklíč a 10/12/14 subklíčů pro každou z iterací. Samotné generování subklíčů využívá řadu operací, mezi nimiž je například kruhový posuv a substituce po bajtech, operace $\oplus$ s danými iteračními konstantami (polynomy s mocninami dvou), rekurzivní operace $\oplus$ s předchozími subklíči, kdy se první subklíč inicializuje pomocí původního klíče ad.

AES nachází využití na mnoha místech, kromě šifrování je vhodný i pro konstrukci hashovacích funkcí, nebo generování pseudonáhodných čísel. Existují různé modifikace, které zvyšují odolnost proti různým druhům útoků, jednou z nich je úprava generátoru klíčů pro zlepšení statistické distribuce pixelů v obrazu [42].

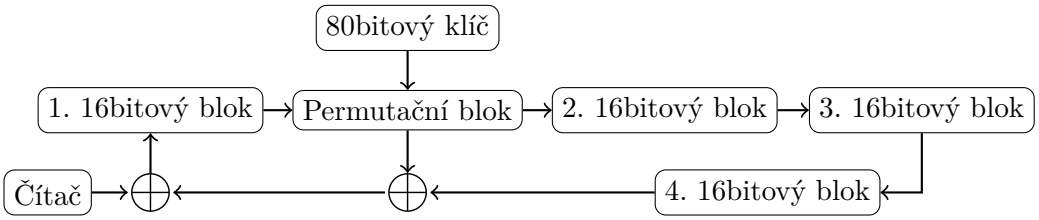


Obrázek 2.5: Schéma šifrování (vlevo) a dešifrování (vpravo) pomocí AES.

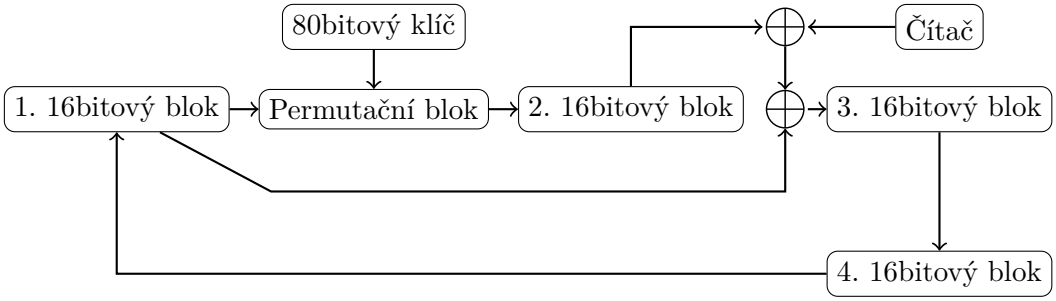
2.1.3 Skipjack

Skipjack je bloková šifra navržená NSA a standardizována [23] organizací NIST pod názvem *Escrowed Encryption Standard* (EES) v roce 1994 v nyní již stáhnuté publikaci FIPS 185. Specifikace algoritmu byla zveřejněna v roce 1998 [25] a v roce 2002 bylo upraveno uspořádání bajtů [30] v blocích algoritmu. Využívá 80bitový klíč a 64bitový blok dat. Šifrování a dešifrování se liší pouze směrem bitového posunu, celkově proběhne 32 iterací, kdy se po osmi iteracích střídají dva režimy zvané Krokovací pravidlo A a B. V obou režimech se nacházejí čtyři datové bloky po 16 bitech, čítač a permutační blok závislý na klíči.

Při šifrování se v režimu A v jedné iteraci první blok permutuje do druhého bloku, druhý blok se přesune do třetího bloku, třetí blok do čtvrtého bloku a do prvního bloku se dostane výsledek operace $\oplus$ nad hodnotou čítače, permutovaným prvním blokem a čtvrtým blokem. Na konci iterace se inkrementuje čítač. V režimu B se v jedné iteraci do druhého bloku opět permutuje první blok, do třetího bloku se uloží výsledek operace $\oplus$ nad prvním blokem, druhým blokem a hodnotou čítače, třetí blok se přesune do čtvrtého bloku a čtvrtý blok do prvního bloku. Na konci iterace se opět inkrementuje čítač. Permutace je invertibilní a využívá určené hexadecimální substituční tabulky pro iterační funkci ve Feistelově struktuře. Při dešifrování se začíná režimem B, obrací se přesouvání bloků, dekrementuje se čítač a permutace je invertována. Strukturu obou režimů lze vidět na Obrázcích 2.6 a 2.7.



Obrázek 2.6: Krokovací pravidlo A u šifry Skipjack ve směru šifrování.



Obrázek 2.7: Krokovací pravidlo B u šifry Skipjack ve směru šifrování.

2.1.4 Módy operací blokových šifer

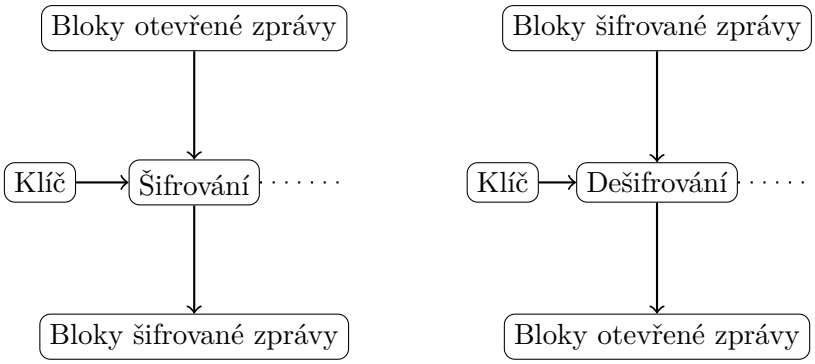
Většina zpráv má délku, která neodpovídá délce bloku očekávané danou blokovou šifrou. Zpráva je často delší a potřebujeme ji rozdělit vhodným způsobem na jednotlivé bloky, které jsou vstupem zvolené šifry. Poslední blok je obvykle také zapotřebí zarovnat na daný počet bitů včetně uložení informace o počtu přidaných bitů, nejčastěji nul. Obvykle toho docílíme uložením čísla do posledního bajtu šifrované zprávy. Další možností je tzv. *Ciphertext Stealing* (CTS), která na rozdíl od zarovnání nezvyšuje délku zprávy. Toho je obvykle dosaženo úpravou módu při zpracovávání posledních dvou bloků zprávy, které se tak stanou vzájemně závislými bez ohledu na konkrétní mód.

Z tohoto důvodu potřebujeme módy operací blokových šifer, které iterativně zpracovávají zprávu po blocích o požadované velikosti a zapouzdřují zvolenou blokovou šifru. V současnosti je organizací NIST schváleno šest módů založených na důvěryhodnosti [29, 32]:

- *Electronic Code Book* (ECB) - schválen roku 1980 [22]
- *Cipher Block Chaining* (CBC) - schválen roku 1980 [22]
- *Cipher Feedback* (CFB) - schválen roku 1980 [22]
- *Output Feedback* (OFB) - schválen roku 1980 [22]
- *Counter* (CTR) - schválen roku 2001 [29]
- XTS-AES - schválen roku 2010 [32]

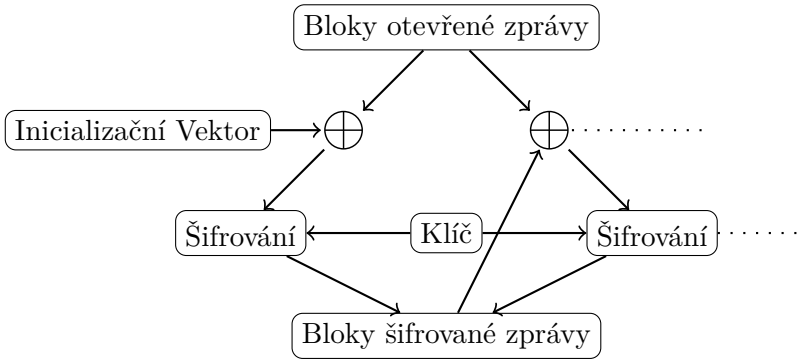
Existují ještě další neoficiální módy operací zaměřené speciálně na šifrování disků, o kterých se zmíníme později v Kapitole 2.2.

ECB mód [22] je ze všech módů nejjednodušší z hlediska realizovatelnosti. Zpráva je rozdělena do bloků o velikosti požadované danou blokovou šifrou, které jsou na rozdíl od ostatních módů nezávisle na sobě (de)šifrovány, což znamená, že se případné chyby nepropagují mezi nimi. Nicméně vzniká bezpečnostní riziko, protože lze sledovat, jak často se opakují šifrované bloky, které se u identických bloků otevřených zpráv neliší. Díky nezávislosti jednotlivých bloků je navíc tento mód plně paralelizovatelný. Schéma šifrování a dešifrování je možné vidět na Obrázku 2.8.

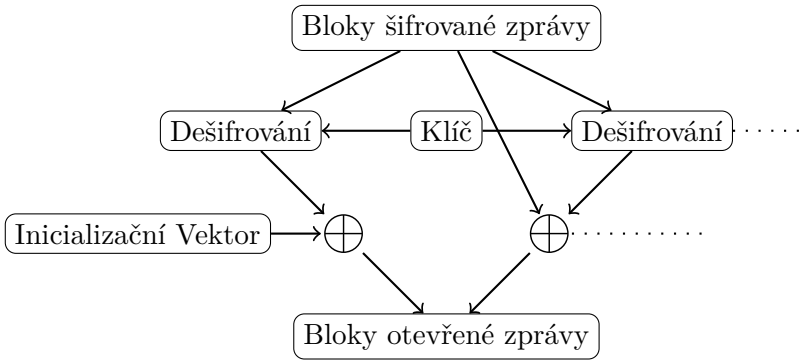


Obrázek 2.8: Schéma šifrování (vlevo) a dešifrování (vpravo) pomocí ECB módu.

CBC mód [22] přináší oproti ECB závislost mezi jednotlivými bloky. Při šifrování se na vstup blokové šifry dostane $\oplus$ bloku otevřené zprávy s předchozím výstupem blokové šifry, kdy na začátku používáme *Inicializační Vektor* (IV), který by měl být náhodný pro zvýšení bezpečnosti. Při dešifrování se naopak nad výstupem blokové šifry provede $\oplus$ s předchozím vstupem blokové šifry, kdy opět používáme na začátku IV. Pokud při dešifrování použijeme špatný IV, chyba nastane pouze u prvního bloku otevřené zprávy, zatímco další bloky budou v pořádku, protože závisejí pouze na předchozím bloku šifrované zprávy. Pokud se objeví chyba v bloku šifrované zprávy, potom ovlivní pouze následující blok otevřené zprávy a na další bloky opět nemá žádný vliv. Protože lze blok otevřené zprávy získat pomocí dvou sousedních bloků šifrované zprávy, je mód při dešifrování paralelizovatelný, což ale neplatí pro šifrování. Na Obrázku 2.9 a 2.10 lze vidět schéma šifrování a dešifrování pomocí CBC.

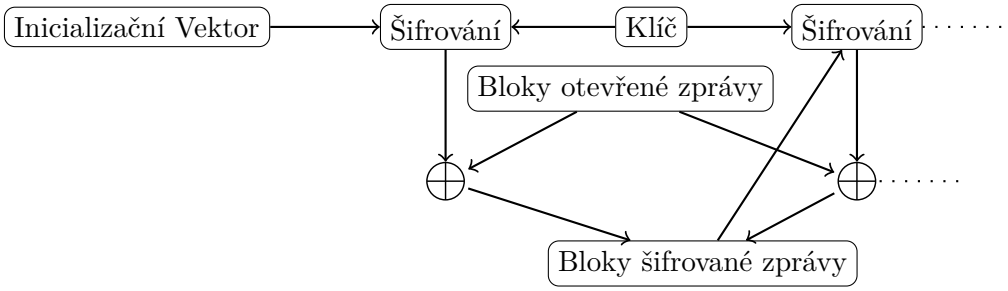


Obrázek 2.9: Schéma šifrování pomocí CBC módu.

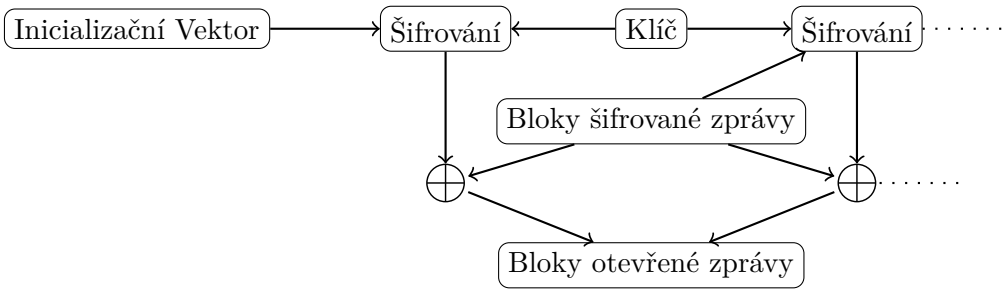


Obrázek 2.10: Schéma dešifrování pomocí CBC módu.

CFB mód [22] je velmi podobný CBC. Zásadním rozdílem je, že zprávu zpracováváme po bitech, takže bloková šifra se celkově chová jako proudová. Při šifrování je vstupem blokové šifry výstup předchozího bloku, nebo Inicializační Vektor (IV) v případě prvního bloku. Nad výstupem blokové šifry se postupně provede operace $\oplus$ s bity otevřené zprávy a výsledek se posílá do vstupního registru v dalším bloku. Při dešifrování pracuje šifra v režimu šifrování, vstupem blokové šifry je buď IV, nebo předchozí blok šifrované zprávy a nad výstupem blokové šifry se volá operace $\oplus$ s šifrovanou zprávou. Stejně jakou u CBC se chyba při dešifrování nepropaguje mezi jednotlivými bloky, ale ovlivní jen určitý počet bitů ve vstupu následujícího bloku. Protože u dešifrování závisí vstup blokové šifry pouze na šifrované zprávě nebo IV, je tento proces paralelizovatelný, ale u šifrování toto neplatí. Schéma šifrování a dešifrování je možné vidět na Obrázcích 2.11 a 2.12.

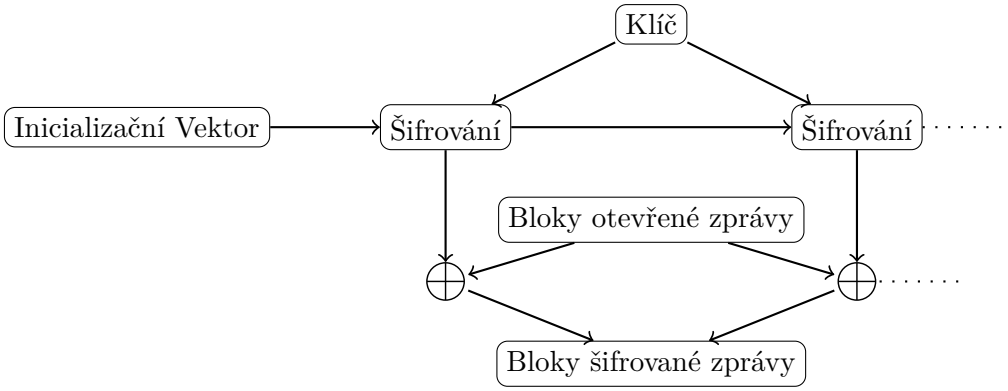


Obrázek 2.11: Schéma šifrování pomocí CFB módu.

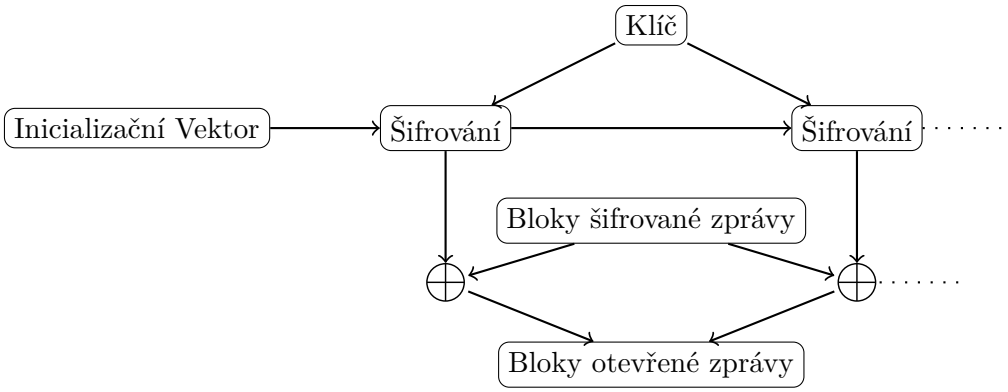


Obrázek 2.12: Schéma dešifrování pomocí CFB módu.

OFB mód [22] podobně jako CFB zpracovává zprávu po bitech a chová se tak jako proudová šifra. Při šifrování je vstupem blokové šifry Inicializační Vektor (IV) v případě prvního bloku nebo předchozí výstup blokové šifry. Výstup blokové šifry se po bitech posílá na vstup do vstupního registru v následujícím bloku a do operace $\oplus$ s bity otevřené zprávy, odkud získáme blok šifrované zprávy. Při dešifrování pracuje bloková šifra opět v režimu šifrování a postup je identický s šifrováním, jen se prohodí bloky otevřené a šifrované zprávy. Propagaci chyb při dešifrování ovlivňuje IV a výstup blokové šifry, takže se často volí opravitelné kódy. Chyba u šifrované zprávy ovlivní pouze konkrétní otevřenou zprávu, ale ztráta této zprávy je nenahraditelná. Protože při šifrování i dešifrování závisí vstup blokové šifry na předchozím výstupu blokové šifry, není tento mód paralelizovatelný. Schéma šifrování a dešifrování je možné vidět na Obrázcích 2.13 a 2.14.

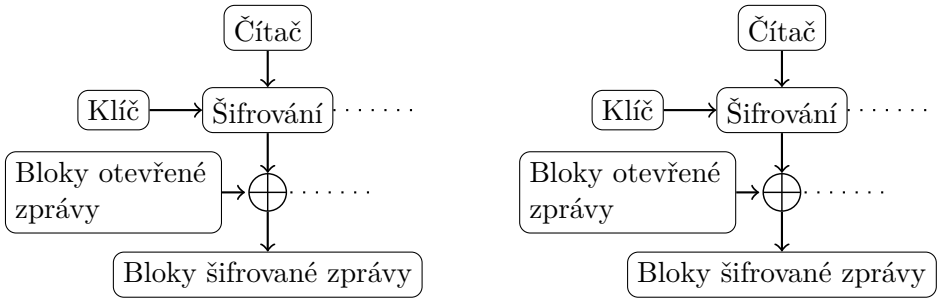


Obrázek 2.13: Schéma šifrování pomocí OFB módu.



Obrázek 2.14: Schéma dešifrování pomocí OFB módu.

CTR mód [29] podobně jako OFB zpracovává zprávu po bitech, ale odstraňuje závislost mezi bloky. Jako vstup blokové šifry používá čítač, který se každou iteraci inkrementuje. Tento čítač se obvykle pomocí vhodné operace kombinuje s dalším číslem, které může být náhodné. Nad výstupem blokové šifry se volá $\oplus$ s otevřenou zprávou. Při šifrování i dešifrování pracuje bloková šifra v režimu šifrování, rozdíl je jen v prohození otevřené a šifrované zprávy. Mód je plně paralelizovatelný, případné chyby se nepropagují. Na Obrázku 2.15 lze vidět schéma šifrování i dešifrování.



Obrázek 2.15: Schéma šifrování (vlevo) a dešifrování (vpravo) pomocí CRT módu.

2.2 Šifrování pevného disku

Tato kapitola čerpá mj. ze studentských prací [18] a [1]. Budeme se zabývat tzv. *Data at Rest*, což jsou persistentní data nacházející se ve zvoleném datovém úložišti, pro naše potřeby konkrétně pevný disk operačního systému Windows. Šifrování dat může probíhat na různých úrovních:

- souborová úroveň (*File-level*)
- adresářová úroveň (*Folder-level*)
- oddílová úroveň (*Partition-level*)
- disková úroveň (*Disk-level*)

My se zaměříme na šifrování na oddílové úrovni a dále jej budeme označovat pod pojmem šifrování disku. Obvyklou metodou je šifrovat data za běhu ještě předtím, než jsou zapsána na datové úložiště a dešifrovat je až tehdy, potřebujeme-li je použít.

Dále rozlišujeme hardwarově a softwarově orientované šifrování, podle vrstvy, na které probíhá, ačkoli v současnosti se oba postupy kombinují. Zatímco hardwarově orientované šifrování dokáže šifrovat po jednotlivých bitech bez ohledu na informace, je považováno za bezpečnější a efektivnější, softwarově orientované šifrování musí brát v úvahu části disku pro zavádění systému a jeho logické dělení (MBR), které musí zůstat nedotknuté, ale je považováno za flexibilnější, levnější a lépe paralelizovatelné. Nás bude zajímat především softwarově orientované šifrování.

Dále rozdělujeme šifrování na tzv. *Narrow-block Encryption* a *Wide-block Encryption*. Narrow-block Encryption šifruje po blocích o obvyklé velikosti 16 bajtů, je rychlejší, ale méně bezpečnější, zatímco Wide-block Encryption šifruje po celých sektorech o obvyklé velikosti 512 bajtů a vyžaduje tak víceprůchodové zpracování, ale v případě dostatečného výpočetního výkonu je vhodnější. My se zaměříme na Narrow-block Encryption.

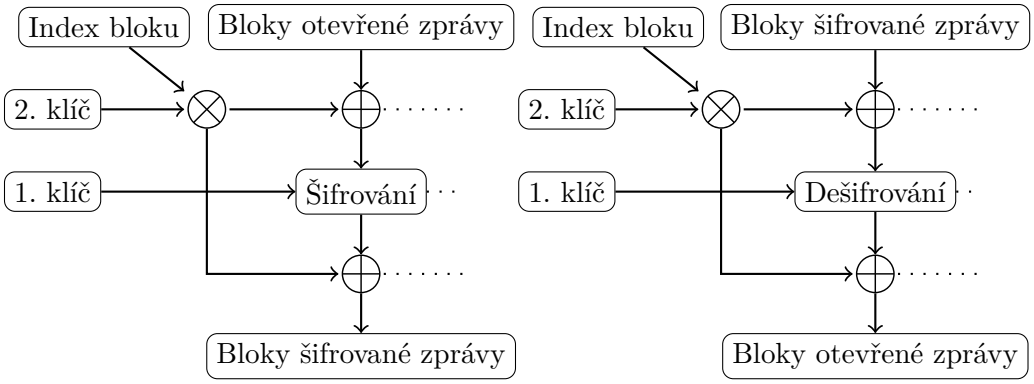
Další možný přístup je dělení z hlediska datové integrity na transparentní a autentizované šifrování, kdy v případě transparentního šifrování není zapotřebí modifikovat uspořádání dat, na rozdíl od autentizovaného šifrování, u kterého po každém šifrovaném bloku následuje kontrolní údaj zajišťující jeho integritu. Nicméně blokově orientovaná zařízení ukládají data do náhodně uspořádaných bloků o konstantní délce, a proto u transparentního šifrování vyžadujeme, aby šifrování bylo nezávislé na jednotlivých blocích dat a na jejich pořadí, a také zachování délky šifrované zprávy vůči původní otevřené zprávě, které neumožňuje dodatečné uložení pomocných hodnot (např. inicializačního vektoru).

Metody softwarově orientovaného šifrování lze rozdělit do dalších tří tříd. Úplné šifrování disku šifruje celý disk kromě MBR, který je zapotřebí k jeho připojení. Při úplném šifrování systémového disku se MBR přesměruje do PBE (pre-boot prostředí), ve kterém dochází k autentizaci uživatele před zpětným dešifrováním disku. Šifrování s virtuálním oddílem vytvoří soubor, se kterým lze po připojení pracovat jako s diskem.

2.2.1 Módy operací blokových šifer

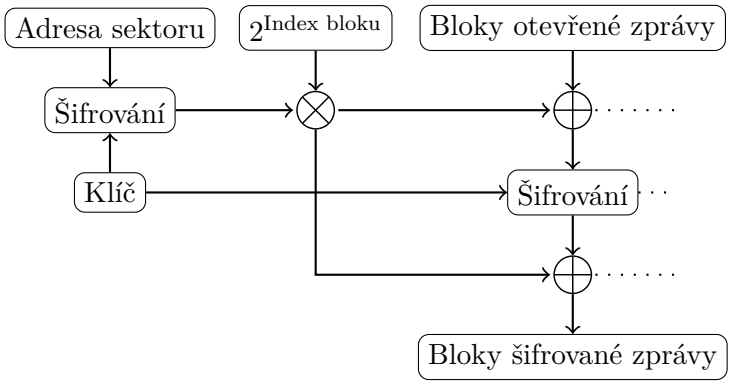
V Kapitole 2.1.4 jsme se zmínili o standardních módech operací pro blokové šifry, které jsou používané v obecných situacích, a nyní se zaměříme na upravené módy operací pro použití u šifrování disků.

V roce 2002 přišla trojice autorů *Liskov, Rivest a Wagner* se studií [17], ve které popisuje obecnou strukturu *Tweakable* módů operací, které se od běžných módů liší dalším vstupem zvaným *Tweak*. Pomocí tohoto nového vstupu obvykle modifikujeme vstup a výstup blokové šifry v jednotlivých iteracích módu. Podle autorů je pojmenovaný *Tweakable Narrow-block* mód operací *LRW*. Tweak se v tomto případě počítá pomocí násobení indexu datového bloku s dodaným 2. klíčem (nikoli 1. klíčem určeným pro blokovou šifru). Tímto způsobem se šifrovaná zpráva naváže na určité místo na disku a chrání před některými útoky založenými na porovnávání stejných bloků. Nicméně vyžadované násobení zpomaluje proces šifrování a vyžaduje heuristiku, např. předpočítání více hodnot pomocí jediného násobení. Nad vstupem a výstupem blokové šifry se poté volá operace $\oplus$ s hodnotou *Tweak*. Šifrování a dešifrování pomocí LRW lze vidět na Obrázku 2.16.

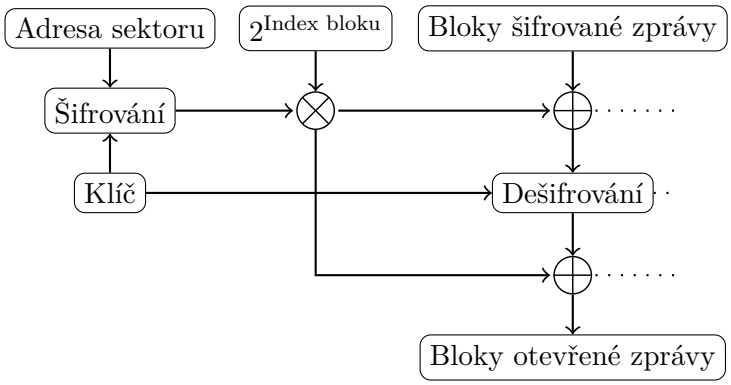


Obrázek 2.16: Schéma šifrování (vlevo) a dešifrování (vpravo) pomocí LRW módu.

Dalším Tweakable Narrow-block módem operací je XEX (*XOR-Encryption-XOR*), jehož autorem je Rogaway [35], který jej použil jako součást jiného módu OCB. XEX lze použít obecně i konkrétně pro šifrování disků. Tweak tvoří násobek zvolené šifrované hodnoty se zvolenými indexy a pro šifrování disků je hodnotou adresa sektoru a dva umocněno na index datového bloku. Operace násobení je tak výrazně jednodušší než u LRW, ale dochází ke dvěma šifrováním pro jeden datový blok, což lze vylepšit předpočítáním šifrované adresy sektoru pro všechny datové bloky ve stejném sektoru. Navíc používá stejný klíč pro šifrování adresy sektoru i datového bloku a nespotřebovává další čas pro generování více klíčů. Tweak se stejně jako u LRW přidá ke vstupu a výstupu blokové šifry. Schémata šifrování a dešifrování lze vidět na Obrázcích 2.17 a 2.18.



Obrázek 2.17: Schéma šifrování pomocí XEX módu.



Obrázek 2.18: Schéma dešifrování pomocí XEX módu.

Existují další Tweakable Narrow-block módy operací, například MCB [7] se podobá XEX, ale využívá samostatné klíče pro Tweak a blokovou šifru a navíc další klíč pro generování masky, která se přidá k hodnotě Tweak a upravuje vstup a výstup blokové šifry.

Mezi Tweakable Wide-block módy operací řadíme například CMC, EME a XCB. CMC [14] podobně jako MCB generuje masku s využitím bloku šifrované zprávy a vytváří tak závislost mezi jednotlivými bloky. EME [13] je na rozdíl od CMC paralelizovatelné a využívá ECB pro úpravu bloku otevřené zprávy. XCB [19] byl navržen zaměstnanci Cisco Systems a využívá tzv. *Luby-Rackoff* strukturu.

XTS-AES (XTS znamená *XEX-based Tweakable CodeBook mode with Ciphertext Stealing*) je další mód založený na důvěryhodnosti, který je navržen speciálně pro šifrování datových úložišť pomocí AES. Vyvinula jej organizace SISWG v roce 2008 [31] a NIST jej schválil v roce 2010 [32]. XTS se chová podobně jako XEX s jedním významným rozdílem, kdy šifrování adresy sektoru a šifrování bloku využívá dva různé klíče na rozdíl od jednoho společného. XTS využívá ECB z hlediska nezávislosti bloků, ale je navíc doplněn o Tweak. Kvůli problému se zarovnáváním bloků, protože AES vyžaduje pevnou velikost 128 bitů, se používá metoda CTS popsaná na začátku Kapitoly 2.1.4.

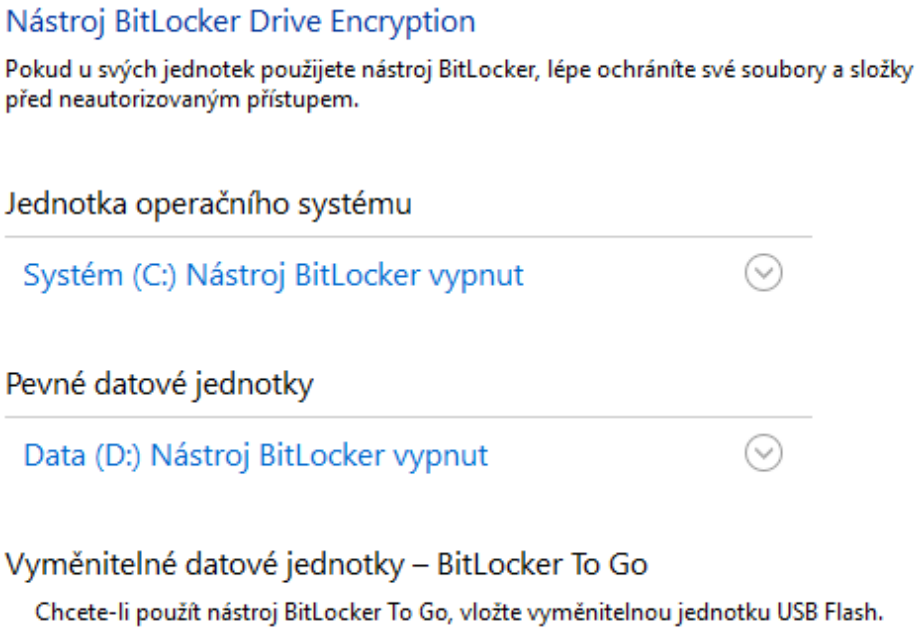
Kapitola 3

Existující nástroje

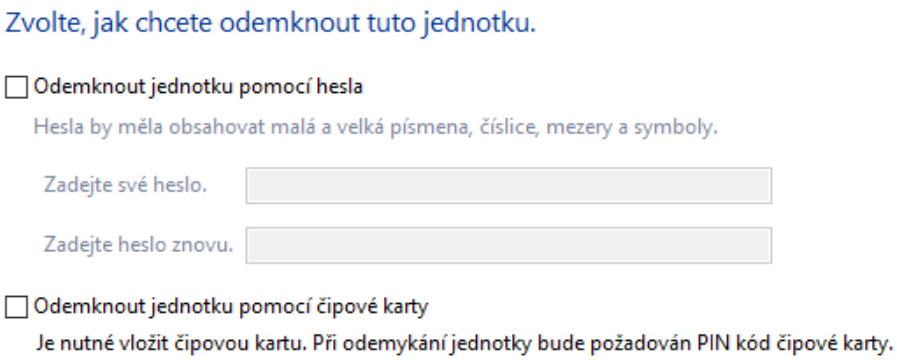
V současnosti existují pro operační systém Windows různé nástroje, placené i volně dostupné, které umožňují šifrování dat. Uvedeme některé z těchto nástrojů (BitLocker, TrueCrypt, FreeOTFE, DiskCryptor, PGP Whole Disk Encryption) a srovnáme jejich vlastnosti a odlišnosti, které šifrovací algoritmy jednotlivé nástroje používají, jejich dostupnost na různých operačních systémech, známé slabiny vůči útokům a ke každému přidáváme dojem z recenze od webového zdroje Techworld, který se již dlouho zabývá analýzou a průzkumem nových technologií v oblasti informačních technologií včetně bezpečnosti.

3.1 BitLocker Drive Encryption

BitLocker [21, 12] je bezplatná funkce integrovaná v operačních systémech Windows Vista a později u vyšších verzí, určená pro úplné šifrování svazků a ochranu dat uložených na svazku s operačním systémem. Od verze Windows 7 navíc umožňuje šifrování externích zařízení po názvem *Bitlocker To Go*, který dále přidal možnost zabezpečení pomocí hesel a čipových karet. Algoritmus v základu používá AES 128 nebo AES 256 v blokovém módu CBC s volitelným difuzorem Elephant. Využívá sofistikovaný systém ukládání klíčů v metadatech a ochrany klíčů např. pomocí TPM, čipových karet, hesel a ochranných klíčů. Funkce umožnuje obnovení dat v případě ztráty hesla i bez zadních dvířek pomocí obnovovacího hesla. Nástroj lze spravovat vzdáleně pomocí WMI nebo příkazového řádku. V případě vyřazování šifrovaných disků lze odstranit kopie klíčů v metadatech a volitelně i v archivech. Kromě transparentního módu využívajícího TPM, který je náchylný na tzv. *Cold Boot Attack* (postavený zachycení dat v paměti kdy je počítač ukončen bez čekání na systém), nabízí ještě autentizační mód PBE prostředí s ochranou pomocí PINu nebo hesla a také mód s USB klíčem, kdy je na USB zařízení uložen soukromý klíč, který po načtení umožňuje spustit systém. V recenzi Techworld [37] je označen za kvalitní nástroj, který prakticky nezatěžuje systém, snadno se používá a je velmi robustní, ale jeho relativně velkou nevýhodou je zmíněná přístupnost pouze od vyšších verzí Windows. Na Obrázcích 3.1 a 3.2 lze vidět ukázky vzhledu funkce ve Windows 10.



Obrázek 3.1: BitLocker Drive Encryption - výběr diskového oddílu.

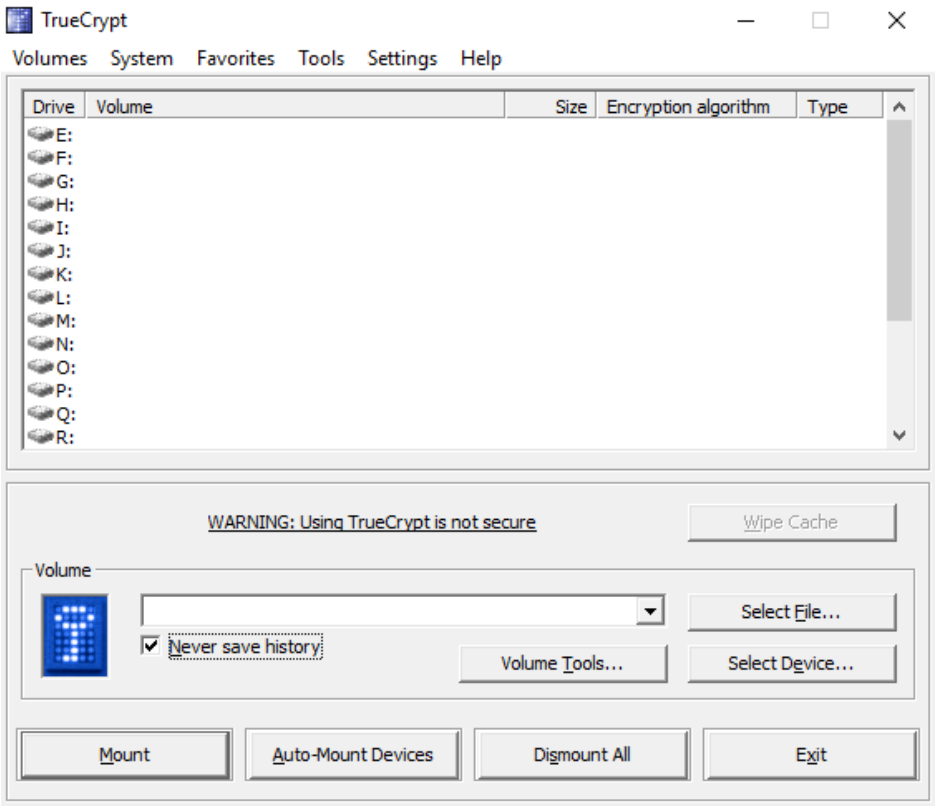


Obrázek 3.2: BitLocker Drive Encryption - parametry šifrování.

3.2 TrueCrypt

TrueCrypt [11] je bezplatný multiplatformní nástroj pro šifrování za běhu s vytvářením virtuálních souborů pro šifrované disky nebo oddíly. V roce 2014 autoři prohlásili, že jeho používání není bezpečné v důsledku množství slabin a vývoj byl zrušen. Nástroj podporuje šifrovací algoritmy AES, Serpent a Twofish, hashovací funkce RIPEMD-160, SHA-512 a Whirpool, umožňuje vytváření skrytých oddílů, které zaručují věrohodnou popiratelnost. V průběhu vývoje se vystřídaly módy operací CBC, LRW a naposledy XTS. Klíče jsou generovány pomocí PBKDF2 s 512bitovou šifrovací solí a 1000 nebo 2000 iteracemi v závislosti na zvolené hashovací funkci. Průběh šifrování lze paralelizovat a vstupně výstupní operace zřetězit pro vyšší výkon. Za bezpečnostní rizika se považuje ukládání klíčů do paměti, možnost zachytávání klávesových vstupů a některé další. Podle recenze Techworld [39] se jednalo

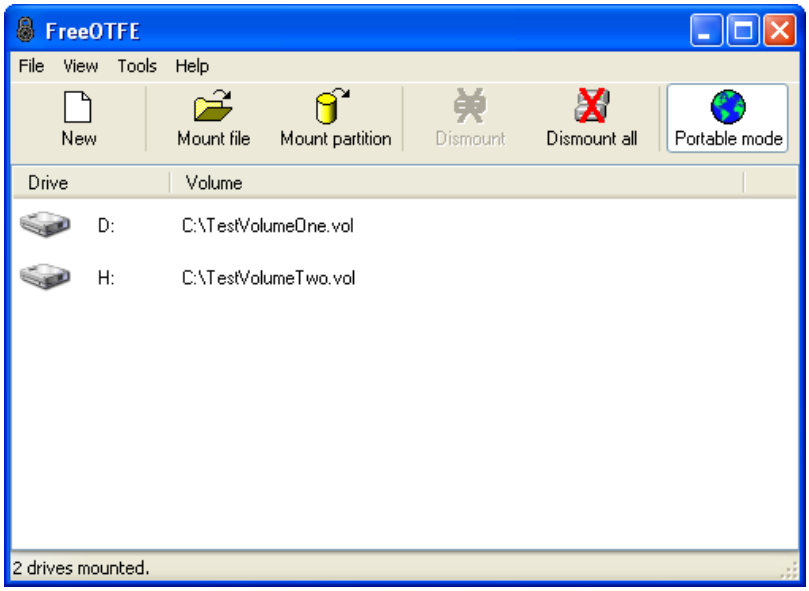
o výborný nástroj po všech stránkách kromě práce se sítěmi, nabízí podrobnou nápovědu a dokumentaci pro používání a byl označen za nejlepší volbu ve své oblasti. Na Obrázku 3.3 lze vidět ukázku vzhledu aplikace ve Windows 10.



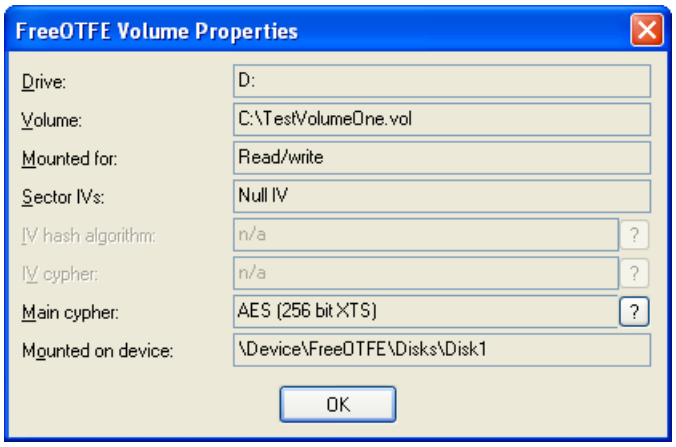
Obrázek 3.3: TrueCrypt - výběr diskového oddílu.

3.3 FreeOTFE

FreeOTFE [10] (OTFE znamená *On The Fly Encryption*) je otevřený nástroj pro šifrování u operačního systému Windows s disky nebo oddíly ve virtuálním souboru. Z jeho názvu vyplývá, že šifrování probíhá za běhu při zápise do virtuálního souboru. Podporuje množství různých hashovacích a šifrovacích algoritmů včetně AES, Blowfish, Serpent a Twofish. Také umožňuje vytvářet skryté oddíly. Podle recenze Techworld [40] je nástroj vhodnou volbou pro uživatele mobilních i klasických Windows, s podporou dodatečných programů pro šifrování i bez administrátorských práv a nabízí více šifrovacích algoritmů než TrueCrypt. Na Obrázcích 3.4 a 3.5 lze vidět ukázky vzhledu aplikace ve Windows XP.



Obrázek 3.4: FreeOTFE - výběr diskového oddílu. Převzato z webu[1].

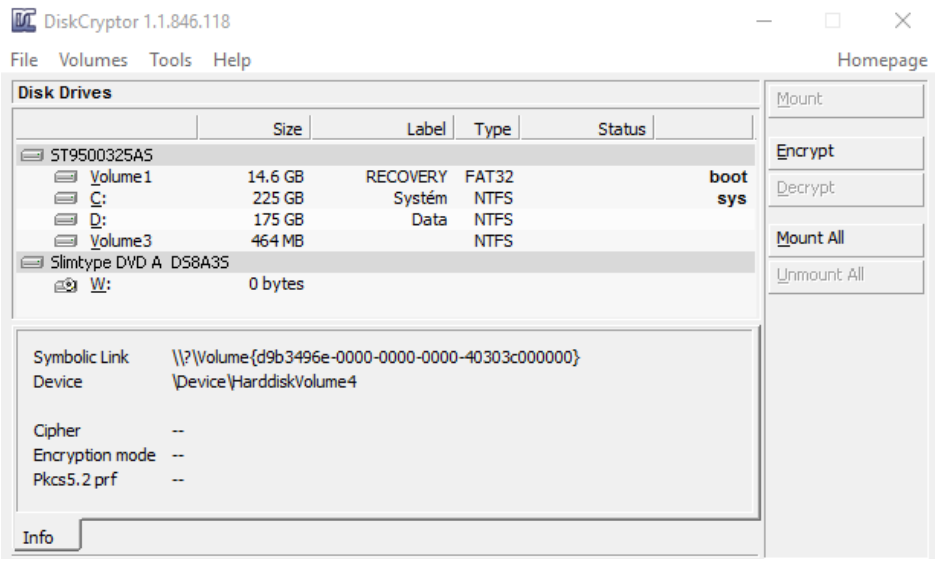


Obrázek 3.5: FreeOTFE - parametry šifrování. Převzato z webu[2].

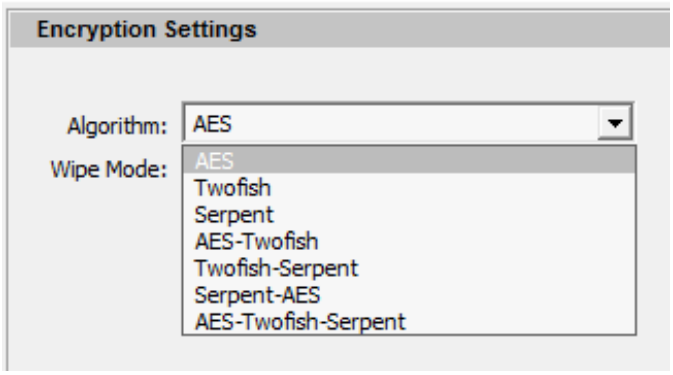
3.4 DiskCryptor

DiskCryptor [9, 6] umožnuje bezplatné transparentní šifrování celého disku nebo jednotlivých oddílů operačního systému Windows pod licencí GNU GPL. Používá šifrovací algoritmy AES, Twofish, Serpent a jejich kombinace v XTS módu. Podporuje také šifrování dynamických disků, externích USB zařízení nebo CD/DVD, poskytuje PBE autentizaci s podporou některých zavaděčů (GRUB, LILO, atd.) a souborů s klíči. Podle recenze Techworld [41] se jedná o užitečný jednoduchý program s drobnými výhradami jako je pomalejší dešifrování a chování správy disků. Na Obrázcích 3.6, 3.7 a 3.8 lze vidět ukázky vzhledu aplikace ve Windows 10.

[1] https://sourceforge.net/projects/freeotfe.mirror/
[2] https://sourceforge.net/projects/freeotfe.mirror/



Obrázek 3.6: DiskCryptor - výběr diskového oddílu.



Obrázek 3.7: DiskCryptor - výběr šifrovacího algoritmu.

Obrázek 3.8: DiskCryptor - zvolení hesla.

3.5 PGP Whole Disk Encryption

PGP Whole Disk Encryption [8, 36], původně pouze PGPDisk, je placený nástroj vyvinutý společností PGP Corporation, která je nyní součástí Symantec. Podporuje operační systémy Windows, Mac OS X a Linux. Umožňuje transparentní šifrování celého disku s autentizací v PBE. Dle požadavků zákaznické firmy je možné načíst disk i bez nutnosti autentizace. Podle informací firmy se jedná o vysoce výkonné šifrování pomocí AES 128 a AES 256 doplněné o proprietární technologie PGP. U operačního systému Windows je možné ověření pomocí čipových karet a TPM. Podle recenze Techworld [38] se jedná o kvalitní flexibilní nástroj, který se používá snadněji než TrueCrypt, výborně dokumentovaný včetně návodů k použití a s možností softwarového upgradu na vyšší verze. Na Obrázku 3.9 lze vidět ukázku vzhledu aplikace ve Windows XP.

Obrázek 3.9: PGP: Whole Disk Encryption - vytvoření virtuálního disku. Převzato z webu[4].

3.6 VeraCrypt

VeraCrypt [16] je otevřený bezplatný multiplatformní nástroj používaný pro tzv. On-the-fly šifrování, kdy není zapotřebí dešifrovat všechna šifrovaná data pro jejich používání. Nástroj je schopný vytvořit soubor s virtuálním šifrovaným diskem, šifrovat oddíl anebo celé datové úložiště navíc s pre-boot autentizací. Kromě standardního oddílu také umožňuje vytvořit skrytý oddíl pro použití v situaci, kdy se útočník dostane k heslu.

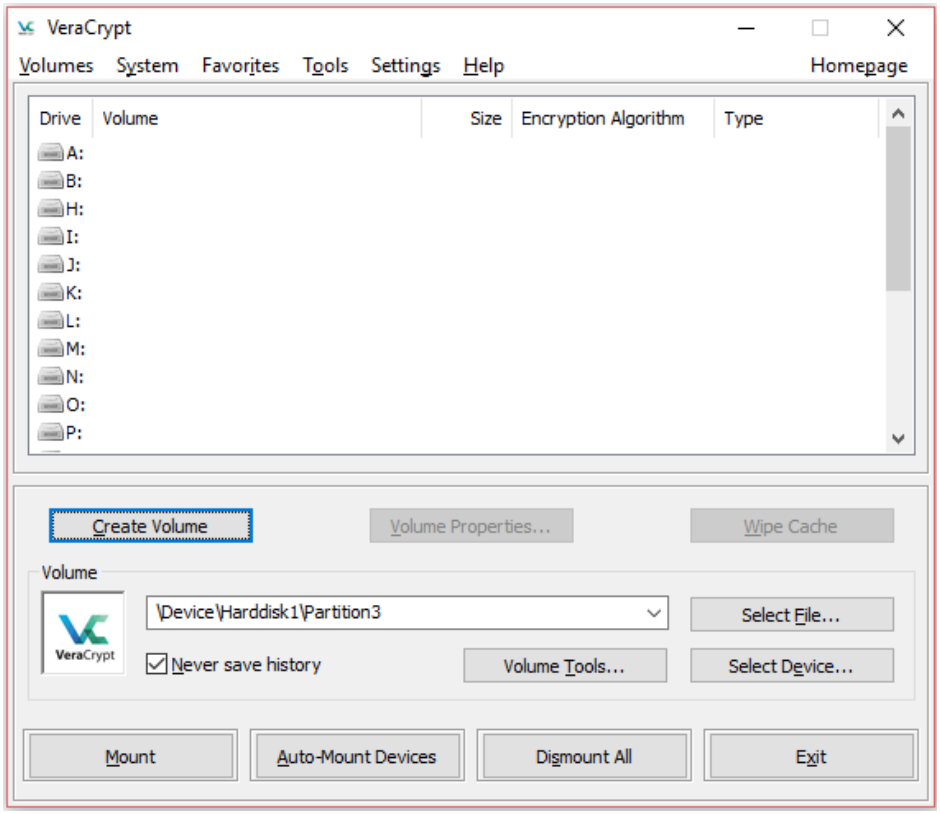
VeraCrypt je nejuznávanější nástupce opuštěného nástroje TrueCrypt, na jehož zdrojovém kódu se zakládá a poskytuje téměř identické uživatelské rozhraní. Šifrování může proběhnout buď s vytvořením a formátováním čistého oddílu (velmi rychlé), nebo postupným šifrováním existujících dat (pomalé, vyžaduje navíc čtení a zápis). Mezi podporované šifrovací algoritmy patří AES-256, Serpent, Twofish a jejich různé kombinace v blokovém módu XTS. Pro generátor pseudonáhodných čísel a odvození šifrovacího klíče (funkce PBKDF2 s 512bitovou kryptografickou solí) se používají hashovací funkce SHA-256, SHA-512 a Whirlpool. Kromě zabezpečení pomocí hesla je možné přidat také soubor libovolného

[4] http://www.pcworld.com/article/127620/article.html

typu, jehož obsah bude kombinován s heslem, a také hodnotu PIM, která určuje počet iterací odvozovací funkce. Entropie náhodných dat je nasbírána z pohybu myši uživatelem. Pro zabránění obnovy šifrovaných dat lze použít některé standardy pro mazání dat, jako jsou tříprůchodový nebo sedmiprůchodový US DoD 5220.22-M, 35průchodový Gutmann nebo pouze jednoprůchodová náhodná data. Šifrování dat lze v průběhu kdykoli přerušit a pokračovat od bodu přerušení i po restartování nebo vypnutí počítače nebo lze dešifrovat data a vrátit změny zpět. Takto šifrovaný disk lze připojit na libovolný neobsazený oddíl a pracovat z daty, která se mezitím nahrávají do RAM paměti. Daný oddíl tak lze používat po zadání správného hesla i bez nutnosti dešifrovat všechna data nebo jej lze trvale dešifrovat. V případě šifrování systémového oddílu lze navíc přidat autentizaci v pre-boot prostředí po spuštění počítače. VeraCrypt podporuje mimo jiné například paralelizaci šifrování na vícejádrových procesorech anebo hardwarově-akcelerované šifrování s AES.

Autoři během vývoje adresovali řadu původních chyb TrueCryptu (ale bez zásadního vlivu na jeho bezpečnost), jejichž seznam lze nalézt v dokumentu [34], mezi které patří například míchání znaménkových a bez-znaménkových datových typů nebo jejich potenciální přetečení a chyby v bootloaderu. Sám VeraCrypt je náchylný na některé neortodoxní útoky [15] jako jsou výše zmíněný Cold Boot Attack, úniky dat v době jejich dešifrování, slabá hesla nebo existence Malware na šifrovaném oddílu.

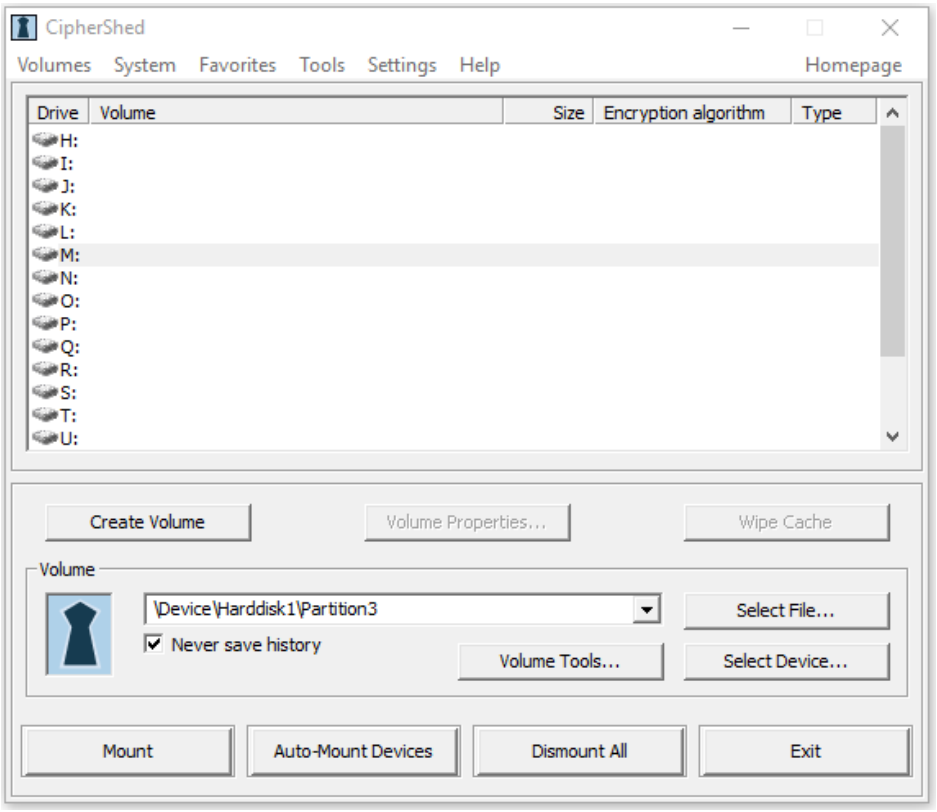
Na Obrázku 3.10 lze vidět hlavní okno aplikace VeraCrypt v operačním systému Windows 10 a jeho podobnost s nástrojem TrueCrypt. V přehledu neobsazených diskových jednotek lze vybrat jednu z nich a připojit k ní šifrovaný oddíl, který se vytvoří prostřednictvím instalačního sekvenčního rozhraní zvaného Wizard.



Obrázek 3.10: VeraCrypt - výběr diskového oddílu.

3.7 CipherShed

CipherShed [3] je otevřený bezplatný multiplatformní nástroj pro On-the-fly šifrování ve vývoji s ukázkovou verzí již k dispozici. Stejně jako VeraCrypt se jedná o následníka nástroje TrueCrypt postaveného na jeho zdrojovém kódu s téměř identickým uživatelským rozhraním. Nástroj oplývá víceméně stejnou funkcionalitou jako výše popsaný VeraCrypt, místo hashovací funkce SHA-256 nabízí zastaralý RIPEMD-160, neumí měnit počet iterací odvozovací funkce klíče a zřejmě zatím nelze zpětně trvale dešifrovat nesystémový oddíl. Na Obrázku 3.11 lze vidět ukázku vzhledu aplikace ve Windows 10 a jeho podobnost s nástrojem TrueCrypt. V přehledu neobsazených diskových jednotek lze vybrat jednu z nich a připojit k ní šifrovaný oddíl, který se vytvoří prostřednictvím instalačního sekvenčního rozhraní zvaného Wizard.



Obrázek 3.11: CipherShed - výběr diskového oddílu.

Kapitola 4

Návrh

V předchozích kapitolách jsme si popsali způsoby šifrování disku, vlastnosti symetrických blokových šifrovacích algoritmů a módy operací těchto šifer navržené přímo pro šifrování disků. Cílem této práce je v jazyce `C/C++/C#` navrhnout, implementovat a otestovat vícejazyčnou aplikaci, která pomocí šifrování diskových oddílů zabezpečí určená data. Aplikace bude využita v praxi při klonování disků. V této kapitole se budeme zabývat jejím návrhem. Základními stanovenými požadavky je šifrování celého disku s operačním systémem Windows a jeho dešifrování pomocí vstupního hesla, jehož bezpečnost bude ověřena.

4.1 Jazyk aplikace

Pro implementaci byl zvolen jazyk `C++`, který lze v případě potřeby snadno kombinovat s assemblerem a na rozdíl od jazyka `C` je flexibilnější, existuje v něm mnoho vysokoúrovňových knihoven, které lze použít pro tuto práci (jmenovitě např. *Crypto++*) a mnohem snázeji se v něm programuje.

Z hlediska překladu bude aplikace v základu podporovat anglický a český jazyk s možností rozšíření. Všechny jazyky budou uložené externě v adresáři `locales` v populárním standardním formátu JSON, přičemž aplikace bude schopna detekovat a načíst tyto soubory a usnadnit tak případné rozšíření o další jazyky.

4.2 Autentizace pomocí hesla

Pro dešifrování disku bude zapotřebí zadat správné heslo, které je zvoleno uživatelem před samotným šifrováním, kdy je požadována určitá úroveň síly bezpečnosti tohoto hesla. Budeme předpokládat délku hesla minimálně 8 znaků a alespoň jeden velký (A-Z), jeden malý (a-z) a jeden numerický (0-9) znak. V případě slabého hesla bude uživatel vyzván, aby zadal nové. Každé heslo bude nutné zadat dvakrát a bude umožněno nechat si jej zobrazit, v opačném případě budou znaky nahrazovány symbolem `*`. Toto heslo se dále zpracovává pomocí nové hashovací funkce *SHA3-256*.

4.3 Metody šifrování

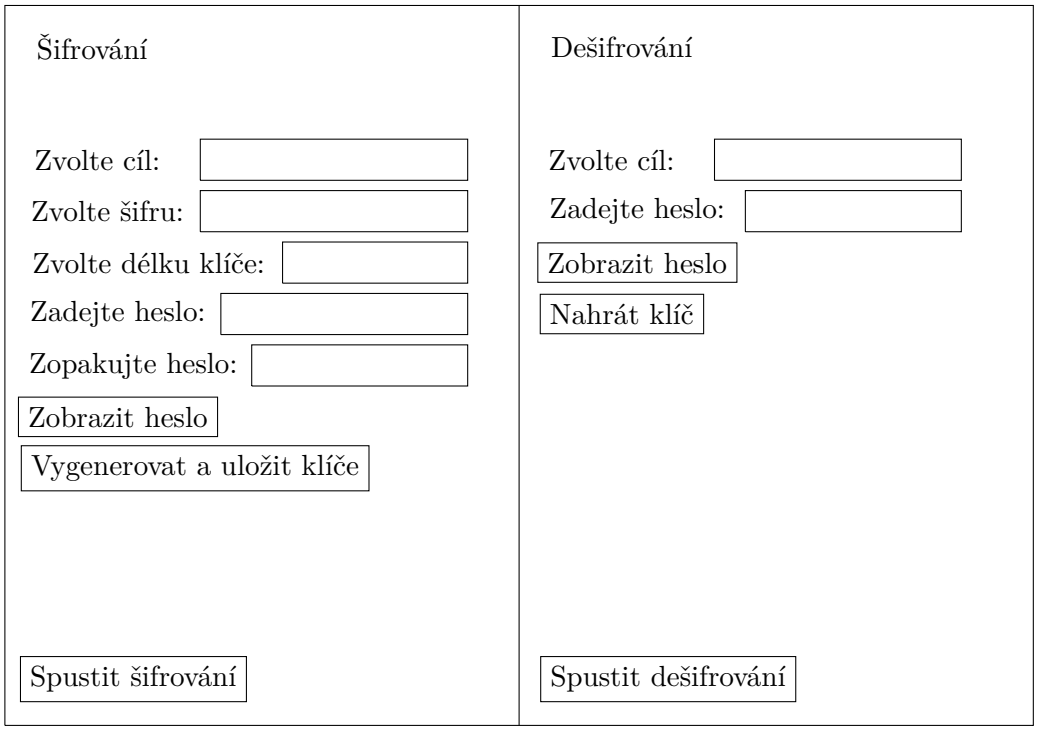
Aplikace bude umožňovat výchozí šifrování pomocí kryptografického standardu AES zvolené délky a také dalších šifer jako jsou Skipjack a 3DES, jejichž implementace je volně dostupná prostřednictvím knihovny *Crypto++*. Kromě šifrovacího algoritmu bude možné

zvolit také délky klíčů, které podporuje zvolená šifra. Ačkoli se největší délka z nabízených považuje za nejbezpečnější, proces šifrování se výrazně zpomaluje. Poté se vygeneruje potřebný šifrovací klíč, který bude možné exportovat a v případě ztráty hesla jej lze použít na obnovení šifrovaných dat.

Jako mód operací nad zvolenou blokovou šifrou použijeme CBC doplněné o CTS, které umožňuje vyhnout se bitovému zarovnávání na velikost bloku daného šifrovacího algoritmu.

4.4 Průběh šifrování a dešifrování

Jakmile jsou v aplikaci zvoleny všechny požadované parametry včetně cílového oddílu, lze spustit proces šifrování. Dešifrování je možné spustit buď nahráním exportovaného klíče, kdy nedochází k žádné kontrole jeho správnosti, anebo pomocí hesla, které naopak umožňuje autentizaci. Na Obrázku 4.1 je návrh grafického rozhraní aplikace.



Obrázek 4.1: Návrh grafického rozhraní.

4.5 Scénář přípravy pro chod aplikace a následné klonování disků

Aplikace bude vyžadovat přítomnost alespoň jednoho nesystémového oddílu, který bude určen pro nahrání dat následované jeho šifrováním. Tento oddíl může být vytvořen i na přenosném médiu. V případě potřeby je možné tento oddíl nejprve zašifrovat pomocí jiného kryptografického nástroje umožňujícího výše zmíněné On-the-fly šifrování, aby se data ani v době před spuštěním aplikace nenacházela na nezabezpečeném úložišti.

Po nahrání dat bude z libovolného místa mimo zvolený oddíl spuštěna aplikace a proběhne šifrování zvoleného oddílu (více informací v Manuálech B a C). Doporučuje se před šifrováním exportovat záložní šifrovací klíč. V průběhu šifrování vznikne v adresáři s aplikací šifrovaný soubor `encrypted_O.BIN`, kde `O` je název diskové jednotky se zvoleným oddílem. Tento soubor společně s heslem jsou klíčovými informacemi pro následné úspěšné dešifrování oddílu.

Následuje klonování disku, jehož výsledkem je vytvoření požadovaného počtu bitových kopií šifrovaného oddílu, které jsou předány zákazníkům. Problematická je distribuce hesla a šifrovaného souboru, popřípadě i záložního klíče. Pro tento případ se obvykle používá asymetrická kryptografie, kdy jsou příjemce i odesílatel mají každý vlastní veřejný a soukromý klíč. Před odesláním jsou data šifrována soukromým klíčem odesílatele (podpis) a veřejným klíčem příjemce v tomto pořadí. Po doručení jsou data dešifrována soukromým klíčem příjemce a veřejným klíčem odesílatele (ověření podpisu) v tomto pořadí. Tak jsou zajištěny všechny bezpečnostní cíle přenosu dat. Tímto způsobem je například možné distribuovat heslo a záložní klíč, zatímco šifrovaný soubor bude předán na přenosném médiu nebo e-mailem (pozor, jedná se o citlivá binární data).

Jakmile je zapotřebí dešifrovat oddíl a data jsou k dispozici, lze v aplikaci importovat záložní klíč nebo použít kontrolní heslo pro dešifrování oddílu. **Pokud se název diskové jednotky s šifrovaným oddílem liší od názvu vyplývajícího ze šifrovaného souboru, je nutné soubor přejmenovat (tato situace může nastat při přenosu)!** Po úspěšném dešifrování soubor zmizí.

Kapitola 5

Implementace

V této kapitole bude popsána samotná implementace aplikace v jazyce C++ s použitím volně dostupné knihovny Crypto++ pro kryptografické operace a rozhraní Windows API pro grafický vzhled.

Základní moduly tvoří třída `DEcore` s jádrem aplikace, prostřednictvím kterého spolu jednotlivé části komunikují, třída `DEcrypto`, která pracuje s knihovnou Crypto++, třída `DEsettings` s metodami pro nastavení jazyka aplikace a třída `DEinterface` pro uživatelské rozhraní Windows API. Vedlejší jsou také hlavičkové soubory pro konstantní hodnoty, zdroje s menu a dialogem a v poslední řadě také JSON parser, jehož autorem je Tomáš Růžička a byl použit s jeho svolením.

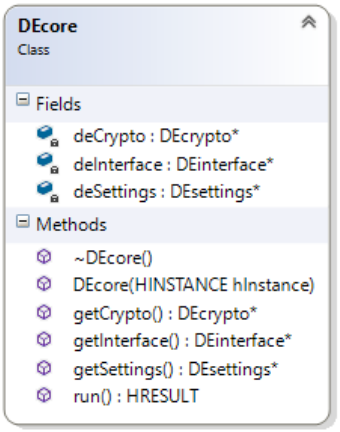
5.1 Knihovna Crypto++

Tato knihovna ve své poslední verzi 5.6.3 byla zvolena především pro svou rozsáhlou funkčnost v oboru kryptografie. Nabízí implementace různých symetrických a asymetrických šifrovacích algoritmů včetně dodržení stanovených standardů, blokové módy operací, hashovací funkce, autentizační kódování, zarovnávání bloků, správu klíčů, generátory pseudonáhodných čísel ad.

Pro uložení šifrovacího klíče a inicializační vektor je zde určena třída `SecByteBlock` pro bezpečnou práci s pamětí. Jako šifrovací algoritmy aplikace využívá dostupné třídy `AES`, `SKIPJACK` a `DES_EDE3` (3DES). Algoritmy jsou vysoce optimalizované a zvláště AES je napsaný v assembleru. Právě AES má k dispozici různé délky klíčů (128, 192 a 256 bitů). Dále po zvážení využívá aplikace strukturu CBC_CTS_MODE, která představuje blokový mód CBC doplněný o metodu CTS popsanou výše umožňující vyhnout se zarovnávání bitů. Knihovna nabízí třídy zvané obecně Source, Sink a Filter, které zjednodušují práci knihovních funkcí s datovými typy C++. Source představuje vstupní data, Sink výstupní data a Filter transformaci mezi nimi. Aplikace využívá pro práci se znakovými poli slouží třídy `ArraySource` a `ArraySink`, pro práci s STL řetězci `StringSource` a `StringSink` a pro převod na hexadecimální kódování znaků `HexEncoder` a `HexDecoder`. Třída `SHA3_256` představuje implementaci nového hashovacího algoritmu SHA3 o 256 bitech. Generování pseudonáhodných čísel probíhá pomocí funkce `OS_GenerateRandomBlock`, která umožňuje blokující nebo neblokující čekání na dostatečnou entropii náhodných dat.

5.2 Třída DEcore

Základní třídou, která se stará o vytvoření a inicializaci všech ostatních tříd, je `DEcore` s jedinou metodou `run`. Tato rodičovská třída vytváří a uchovává si objekty ostatních tříd `DEsettings`, `DEcrypto` a `DEinterface` a právě skrze ni zpřístupňuje komunikaci mezi nimi. Po spuštění aplikace dojde k načtení předpokládaných jazykových souborů s příponou JSON z adresáře `locales` umístěného u aplikace. Aplikace pro svůj běh vyžaduje alespoň jeden jazyk. Čeština má prioritu před angličtinou a angličtina před ostatními jazyky. Po načtení jazyků se inicializuje uživatelské rozhraní a spustí smyčka správ Windows API.



Obrázek 5.1: Diagram třídy DEcore.

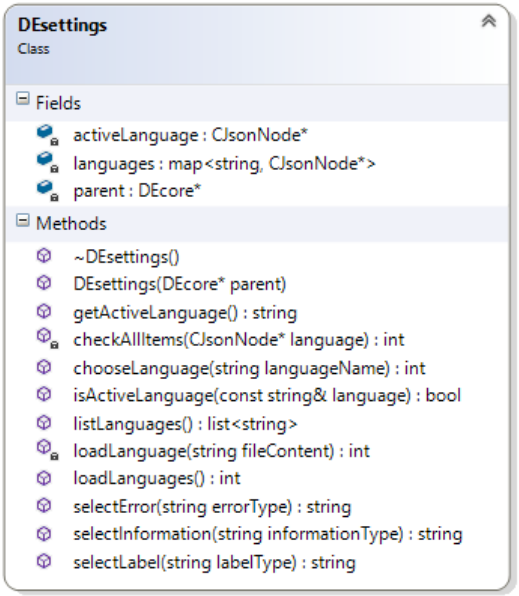
5.3 Třída DEsettings

Třída, která obstarává vícejazyčnost aplikace, se nazývá `DEsettings`. Při načtení jazyků metodou `loadLanguages` je procházen adresář `locales` a všechny soubory s příponou JSON v něm jsou postupně prozkoumány metodou `loadLanguage`. Pokud se v těchto souborech vyskytují chyby, které neprojdou přes JSON parser anebo se pomocí metody `checkAllItems` zjistí chybějící položky, které aplikace vyžaduje, jazyk nebude přidán do aplikace.

Jednotlivé jazyky jsou uchovávány ve vnitřní stromové struktuře `CJsonNode` a pomocí STL kontejneru `map` přiřazeny ke svých názvům v proměnné `languages`, které se předpokládají být unikátní a následující duplicitní jazyky nejsou přidány. Metody třídy umožňují také zjistit (`getActiveLanguage`, `isActiveLanguage`) nebo nastavit (`chooseActiveLanguage`) aktivní jazyk aplikace, vyhledat text, který aplikace v dané chvíli požaduje (`selectError`, `selectInformation`, `selectLabel`) a vypsat všechny jazyky k dispozici (listLanguages).

5.4 Třída DEcrypto

Třída zprostředkující rozhraní knihovny Crypto++ je `DEcrypto`. Pro práci s daty předávanými z uživatelského rozhraní byl zvolen datový typ STL `vector` se šablonou `byte`, což je neznaménkový `char`. Kromě výše zmíněné metody pro generování bloku s pseudonáhodnými čísli a zvolení konstantního inicializačního vektoru potřebné délky poskytuje tato třída také následující funkce pro převod mezi datovými typy a kódováními: `binary2block`



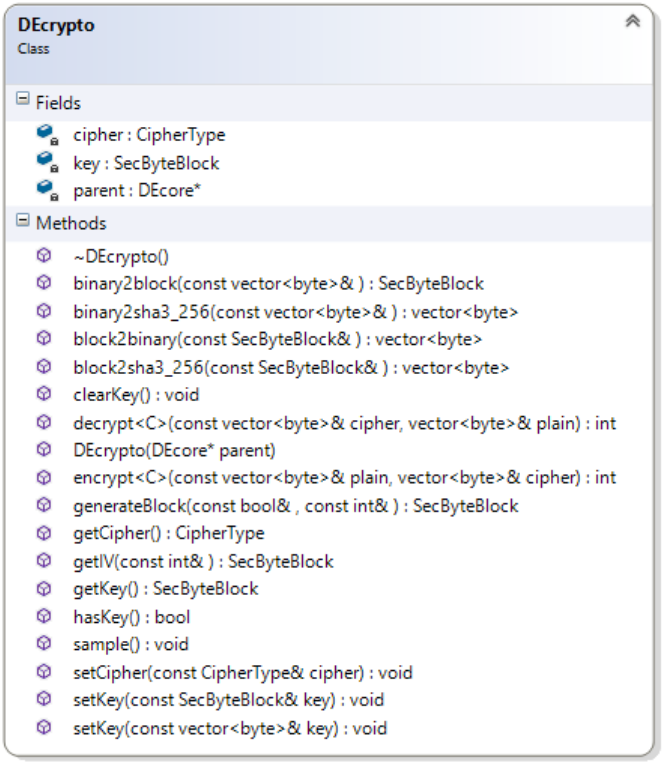
Obrázek 5.2: Diagram třídy DEsettings.

pro `SecByteBlock`, `block2binary` pro binární data a `binary2sha3_256` a `block2sha3_256` pro SHA3-256 hashování.

Třída má dva členy, `key` pro šifrovací klíč a `cipher` se zvoleným šifrovacím algoritmem a k nim poskytuje standardní gettery a settery. Datový typ šifrovacího algoritmu společně s možnými délkami klíčů, délkami datových bloků pro jednotlivé algoritmy a konstantní hodnotou inicializačního vektoru jsou definovány v externím hlavičkovém souboru `DEtypes.h`. Hlavními metodami třídy jsou `encrypt` a `decrypt` pro šifrování a dešifrování binárních dat v tomto pořadí. Obě operace jsou napohled téměř stejné, jen pro transformaci vstupních dat ve struktuře `ArraySource` na výstupní data ve struktuře `ArraySink` za pomocí filtru `StreamTransformationFilter` využívají buď `CBC_CTS_Mode<C>::Encryption`, nebo `CBC_CTS_Mode<C>::Decryption`, kde `C` je třída s šifrovacím algoritmem.

5.5 Třída DEinterface

Nejrozsáhlejší třída pro uživatelské rozhraní Windows API se nazývá `DEinterface`. V tomto modulu se nachází také struktura `threadData` s daty pro vlákna, ve kterých se spouští časově náročné operace čtení a zápisu z disku při šifrování a dešifrování. V této struktuře se předává velikost bufferu pro data, velikost oddílu, ukazatelé na grafické prvky s údaji o postupu šifrování, ukazatel na třídu `DEinterface` a název souboru s údaji potřebnými k šifrování. Třída obsahuje členy `appState` pro rozlišení mezi okny šifrování a dešifrování, `partitionHandle` pro práci se zvoleným oddílem, `hInstance` identifikující proces aplikace pro potřeby Windows API, `wndClassEx` pro registraci hlavního okna uživatelského rozhraní, objekty oken dialogu s průběhem operace (`hWndProgressDialog`), dialogu s výběrem jazyka aplikace (`hWndLangDialog`) a hlavního okna (`hWndMain`), stavové proměnné `revEnc` a `revDec` pro signalizaci zrušení operace a potřebného návratu změn a poslední STL `map` kontejner `partitionInfo`, kam se nahrávají textové informace o viditelných oddílech při spuštění aplikace.



Obrázek 5.3: Diagram třídy DEcrypto.

Většina členů má opět ve třídě své gettery a settery, informační a chybové výpisy řeší funkce `showInfo` a `showError` v tomto pořadí, metoda `runMessageLoop` obstarává smyčka správ Windows API a pomocí `initialize` je registrováno a načteno hlavní okno aplikace, nastaven font písma, jazyk prvků menu (`translateMenu`), pomocí metody `preloadPartitionInfo` načteny informace o oddílech (pozor: tato operace může trvat až jednotky sekund), a nakonec jsou vytvořeny prvky pro podokna pro šifrování, které je v aplikaci výchozí.

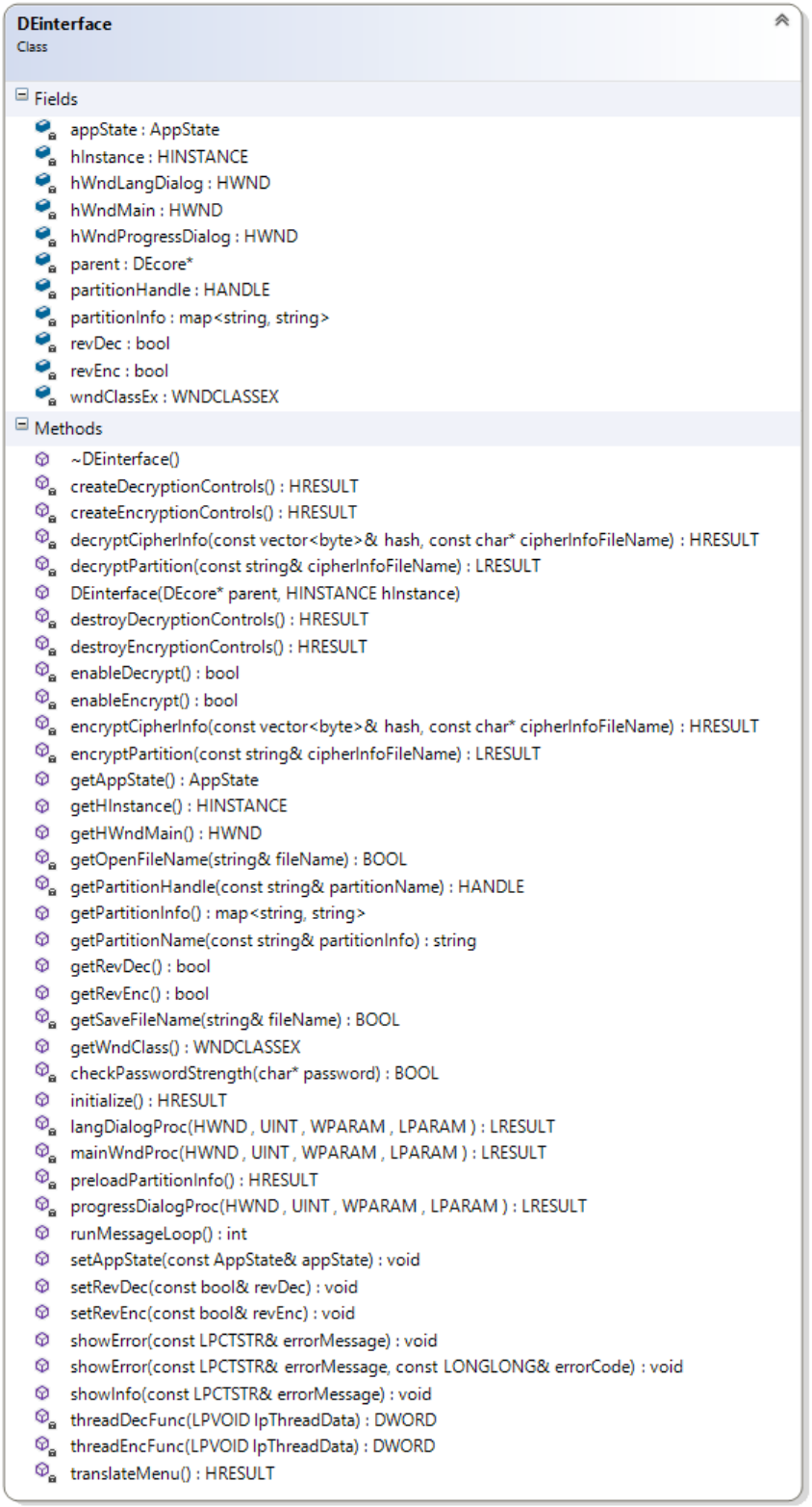
Metody `createDecryptionControls` a `createEncryptionControls` slouží k vytvoření prvků podoken pro dešifrování a šifrování. Společnými prvky jsou výběr cílového oddílu, možnost importování klíče, zadání hesla, přepínač mezi zobrazenými a utajenými znaky hesla a tlačítko pro spuštění dané operace. Šifrování má navíc výběr šifrovacího algoritmu, možnost vygenerovat a exportovat klíč a nutnost ještě jednou zopakovat zadané heslo. Soubor s importovaným klíčem obsahuje první bajt s typem šifrovacího algoritmu a následuje samotný klíč. Formát je mírně podobný souboru s údaji o šifrovaném oddílu, který bude popsán dále v textu. Pro kontrolu dostatečné síly bezpečnosti zvoleného hesla slouží metoda `checkPasswordStrength`, funkce `getSaveFileName` a `getOpenFileName` spustí Windows dialog s výběrem souboru a metoda `getPartitionHandle` vrací objekt pro operace nad zvoleným oddílem. Funkce `enableDecrypt` a `enableEncrypt` ověřují, zda je k dispozici šifrovací klíč a v případě šifrování i existenci nelišících se hesel.

Statická metoda `mainWndProc`, tzv. loopback, je určena pro vyhodnocování zpráv, prostřednictvím kterých okno a jeho prvky komunikují se systémem Windows a reagují na interakci uživatele. Nachází se zde obsluha menu, tedy ukončení aplikace, přepínání mezi šifrováním a dešifrováním, informace o aplikaci a nastavení jazyka, které využívá vlastní

dialog s funkcí `langDialogProc` popsaný prostřednictvím externích zdrojů. Při změně jazyka se prvky v okně zruší a znovu vytvoří. Při změně šifrovacího algoritmu je zapotřebí načíst nebo vygenerovat nový klíč, ale je možné načíst libovolný klíč ve výše zmíněném formátu bez ohledu na zvolený šifrovací algoritmus. Vygenerovaný i načtený klíč je možné exportovat. Při zadávání hesla se průběžně kontroluje, zda je již dostatečně silné z hlediska bezpečnosti, přičemž tato informace se nachází vpravo od vstupu a požadavky na heslo je možné zobrazí přejetím myši přes otazník. Upozornění, zda obě zadaná hesla souhlasí, se nachází vpravo od druhého vstupu. Pod vstupy pro heslo je přepínač pro jejich odtajení v případě, kdyby si uživatel chtěl zkontrolovat správnost hesla sám.

Při spuštění šifrování nejprve proběhne příprava. Opět je zkontrolována správnost hesel a přítomnost klíče, načten adresář s aplikací, ve kterém se vytvořím soubor se jménem ve tvaru `encrypted_P.BIN`, kde P je jméno jednotky zvoleného oddílu. Přítomnost tohoto souboru značí, že disk je již zašifrovaný a není možné jej šifrovat znovu. Obsah souboru je téměř totožný s formátem exportovaného klíče, jen mu předchází SHA3-256 hash hesla. Tento soubor je dále pomocí funkce `encryptCipherInfo` šifrován AES-256, kde klíčem je SHA3-256 hash hesla. Poté je získán objekt pro práci se zvoleným oddílem a funkce `encryptPartition` zjistí další nutné informace. Nejprve se zobrazí varování, že disk bude šifrován, takže se lze stále vrátit zpět. Poté je zjištěna velikost oddílu, souborový systém, uživatelský název a také velikosti alokačních jednotek. Dochází k pokusu o uzamčení oddílu pro přístup k němu, které vyžaduje, aby v dané chvíli nebyl používán, tedy se určitě nejedná o oddíl, na kterém běží systém Windows. Z velikosti alokačních jednotek je určena maximální velikost dat, které je možné číst najednou z oddílu, protože musí být zarovnaná na celé sektory. Dále je vytvořen dialog s metodou `progressDialogProc` pro vyhodnocování zpráv, ve kterém bude uveden popis průběhu operace zahrnující textovou i vizuální velikost již šifrovaných dat. Pro tento dialog je vytvořeno samostatné vlákno aplikace s obslužnou funkcí `threadEncFunc`. Zde dochází k samotnému čtení dat ze zvoleného oddílu, zobrazení průběhu v dialogu, jejich šifrování a následný zápis na stejné místo, odkud byly přečteny, což se opakuje, dokud jsou data k dispozici anebo bylo šifrování zrušeno tlačítkem dialogu. V případě přerušení je volána inverzní operace dešifrování nad šifrovanou částí oddílu, kterou již nelze přerušit bez trvalých následků. Po skončení operací je aplikace automaticky ukončena.

Průběh dešifrování je obdobný, nejprve proběhne příprava, ověření existence výše zmíněného souboru s příponou BIN s údaji o šifrování, dále pokud byl načten klíč, není zapotřebí ověřovat zadané heslo, v opačném případě je ve funkci `decryptCipherInfo` soubor dešifrován za pomocí hesla, ověří se hashe zadaného a uloženého hesla, v případě úspěchu se načte šifrovací algoritmus a klíč a následuje volání metody `decryptPartition` pro získání podrobnějších informací o oddílu. Opět je zobrazeno varování, že disk bude dešifrován, jsou získány stejné výše zmíněné informace jako u šifrování, oddíl je uzamčen, zjistí se maximální velikost dat, které je možné číst z oddílu najednou, vytvoří se dialog pro zobrazení průběhu operace a je vytvořeno nové vlákno s obslužnou funkcí `threadDecFunc`. Podobně jako u šifrování se postupně čtou, dešifrují a zapisují data, dokud jsou některá k dispozici nebo byla operace přerušena, přičemž tehdy je spuštěna inverzní operace šifrování dešifrované části oddílu. Aplikace je automaticky ukončena po skončení operace. Soubor s údaji o šifrovaném disku je v případě chyby nebo zrušení šifrování odstraněn, podobně jako v případě chyby nebo zrušení dešifrování ponechán.



Obrázek 5.4: Diagram třídy DEinterface.

Kapitola 6

Vyhodnocení

V této kapitole bude otestována závislost délky trvání operací šifrování a dešifrování na velikosti oddílu, zvoleném šifrovacím algoritmu a délce šifrovacího klíče. Šifrování oddílů bude probíhat na interním a v prvním případě i externím harddisku, což prokáže také výrazný dopad rychlosti přístupu k pevnému disku se zvoleným oddílem. Budou testovány všechny aplikací podporované šifrovací algoritmy (AES-128, AES-192, AES-256, 3DES, Skipjack) v blokovém módu CBC s CTS na oddílech o zvolených velikostech. Čtení dat v aplikaci obstarává Windows API metoda `ReadFile` a zápis metoda `WriteFile`. Měření rychlosti probíhá s pomocí metod Windows API s vysokou přesností ($< 1\,\mu$s) `QueryPerformanceCounter` a `QueryPerformanceFrequency`.

6.1 Průběh testování na první sestavě

První testovací sestavou je notebook ASUS N61VG-JX086V s procesorem Intel Core 2 Duo T6600 2.20 GHZ, pamětí RAM 4 GB a 64bitovým operačním systémem Windows 10 Education. Interní 2.5palcový pevný disk ST9500325AS má 500GB a 5400 otáček/min. Externí 3.5palcový pevný disk WDC WD800AAJB připojený přes USB konektor má 80 GB a 7200 otáček/min. Aplikací daná velikost bufferu dat je 82 252 800 B ~ 78 MB.

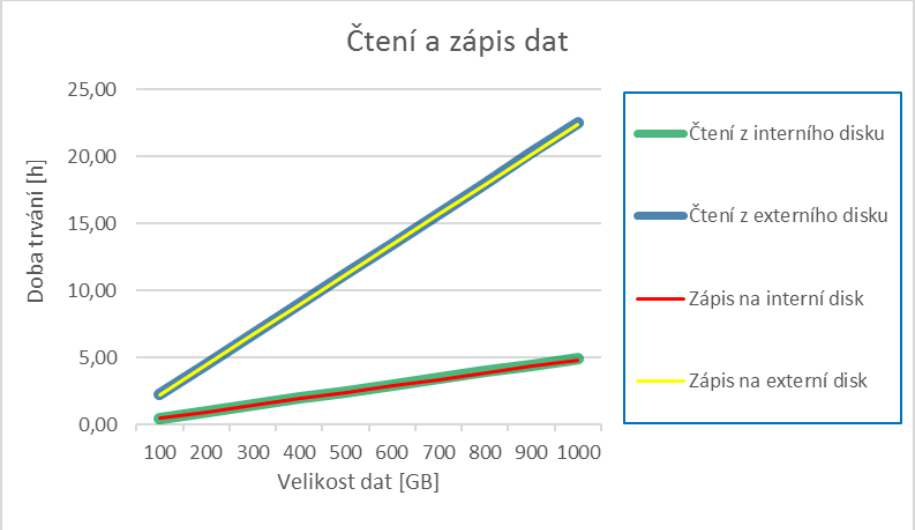
V Tabulce 6.1 lze vidět následující výsledky měření čtení a zápisu dat na interní i externí výše zmíněný pevný disk: změřenou dobu trvání pro 78 MB (méně je lépe), její průměrnou hodnotu, dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.1 je zobrazena závislost doby trvání čtení a zápisu dat na velikosti dat od 100 GB do 1 TB.

V Tabulce 6.2 je možné vidět následující výsledky měření samotného šifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack: změřenou dobu trvání pro 78 MB (méně je lépe), její průměrnou hodnotu, dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.2 je zobrazena závislost doby trvání šifrování dat na velikosti dat od 100 GB do 1 TB.

V Tabulce 6.3 je možné vidět následující výsledky měření samotného dešifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack: změřenou dobu trvání pro 78 MB (méně je lépe), její průměrnou hodnotu, dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.3 je zobrazena závislost doby trvání dešifrování dat na velikosti dat od 100 GB do 1 TB.

Čtení z pevného disku		Zápis na pevný disk	
Interní	**Externí**	**Interní**	**Externí**
Změřená doba trvání pro 78 MB [s]			
1,30	6,20	1,43	6,18
1,31	6,22	1,31	6,17
1,42	6,20	1,30	6,20
1,46	6,20	1,31	6,18
1,34	6,19	1,33	6,17
1,37	6,19	1,32	6,17
Průměrná doba trvání pro 78 MB [s]			
~1,37	~6,20	~1,33	~6,18
Doba trvání pro 100, 500 a 1000 GB [h]			
~0,50	~2,25	~0,48	~2,24
~2,48	~11,24	~2,42	~11,20
~4,95	~22,49	~4,83	~22,41
Průměrná rychlost [MB/s]			
~57	~13	~59	~13

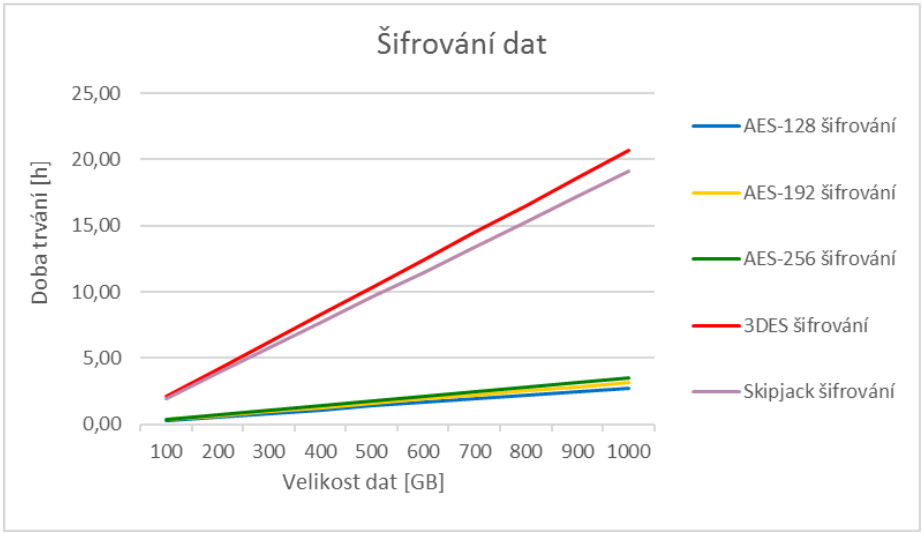
Tabulka 6.1: Výsledky měření parametrů pevných disků.



Obrázek 6.1: Závislost doby trvání čtení a zápisu dat na velikosti dat.

Šifrovací algoritmy v režimu šifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 78 MB [s]				
0,75	0,96	1,02	5,72	4,64
0,75	0,87	0,95	5,69	4,83
0,75	0,84	1,00	5,72	4,95
0,77	0,82	0,96	5,70	5,56
0,73	0,83	0,96	5,86	5,96
0,79	0,87	0,93	5,49	5,70
Průměrná doba trvání pro 78 MB [s]				
~0,75	~0,87	~0,97	~5,70	~5,27
Doba trvání pro 100, 500 a 1000 GB [h]				
~0,27	~0,31	~0,35	~2,07	~1,91
~1,37	~1,57	~1,76	~10,33	~9,56
~2,74	~3,14	~3,51	~20,66	~19,12
Průměrná rychlost [MB/s]				
~104	~91	~81	~14	~15

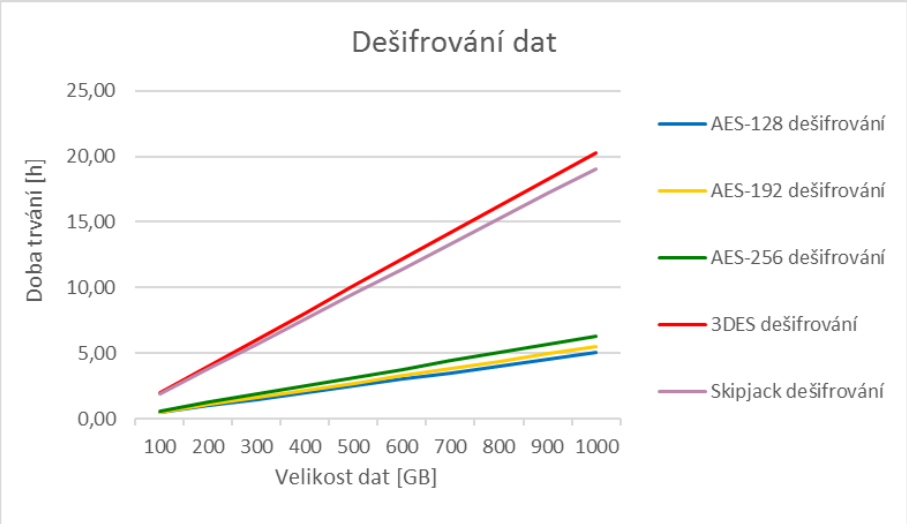
Tabulka 6.2: Výsledky měření šifrování oddílů.



Obrázek 6.2: Závislost doby trvání šifrování dat na velikosti dat.

Šifrovací algoritmy v režimu dešifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 78 MB [s]				
1,42	1,47	1,75	5,59	6,69
1,41	1,52	1,73	5,58	4,71
1,43	1,55	1,71	5,56	4,88
1,33	1,54	1,73	5,62	4,74
1,41	1,54	1,81	5,59	5,22
1,36	1,50	1,75	5,60	5,31
Průměrná doba trvání pro 78 MB [s]				
~1,39	~1,52	~1,75	~5,59	~5,26
Doba trvání pro 100, 500 a 1000 GB [h]				
~0,50	~0,55	~0,63	~2,03	~1,91
~2,52	~2,75	~3,17	~10,14	~9,53
~5,05	~5,51	~6,34	~20,27	~19,06
Průměrná rychlost [MB/s]				
~56	~52	~45	~14	~15

Tabulka 6.3: Výsledky měření dešifrování oddílů.

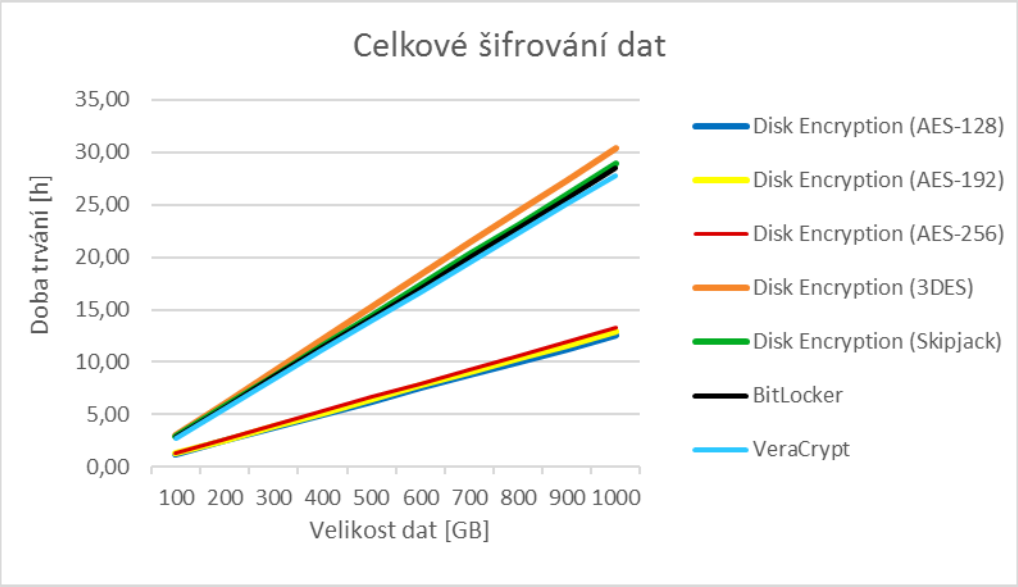


Obrázek 6.3: Závislost doby trvání dešifrování dat na velikosti dat.

V Tabulce 6.4 je možné vidět výsledky měření doby trvání celkového šifrování dat pro 100, 500 a 1000 GB (méně je lépe) a průměrnou rychlost (více je lépe) u nástrojů BitLocker a VeraCrypt ve srovnání šifrovacími algoritmy podporovanými touto aplikací. Nástroj BitLocker, otestován v blokovém režimu XTS, je schopen také detekovat volné místo na disku a přeskočit jej, což výrazně urychlí danou operaci, ale pro účely testování byla tato vlastnost nevyužita. Pomocí nástroje VeraCrypt byl vytvořen standardní VeraCrypt oddíl, data byla šifrována postupně, byly zvoleny algoritmy AES a SHA-512, zadána obyčejné heslo a odmítnut režim bezpečného mazání dat. Hodnoty u šifrovacích algoritmů této aplikace představují celkovou dobu trvání operace (čtení, šifrování a zápis). Na Obrázku 6.4 je zobrazena závislost doby trvání šifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací nástroje v režimu šifrování						
BitLocker	**VeraCrypt**	**AES-128**	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Doba trvání pro 100, 500 a 1000 GB [h]						
~2,84	~2,78	~1,25	~1,29	~1,33	~3,04	~2,89
~14,22	~13,91	~6,26	~6,46	~6,65	~15,22	~14,45
~28,44	~27,81	~12,52	~12,92	~13,30	~30,44	~28,91
Průměrná rychlost [MB/s]						
~10	~10	~23	~22	~21	~9	~10

Tabulka 6.4: Srovnání šifrovacích nástrojů v režimu šifrování.

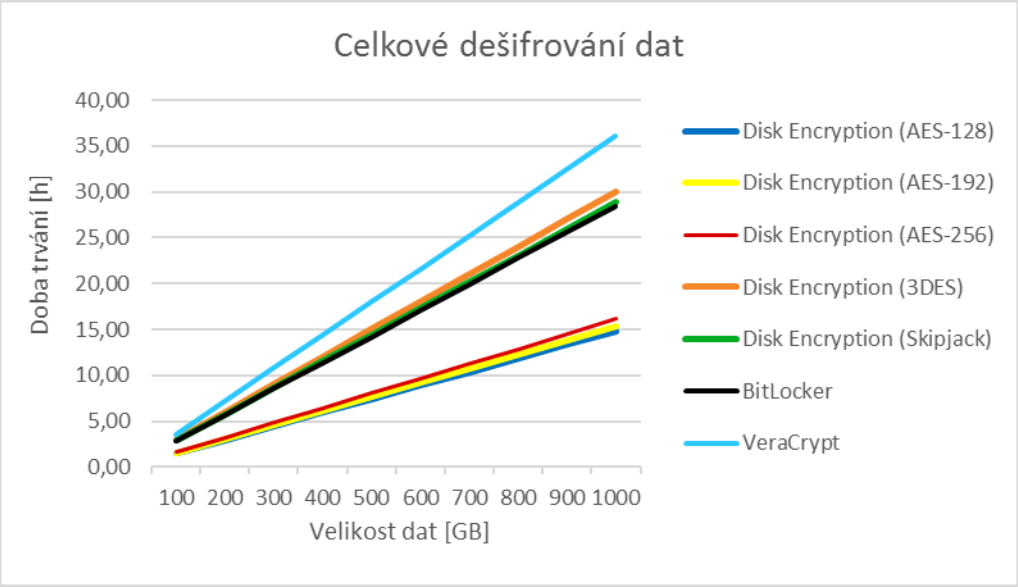


Obrázek 6.4: Srovnání šifrovacích nástrojů v režimu šifrování.

V Tabulce 6.5 je možné vidět výsledky měření doby trvání celkového dešifrování dat pro 100, 500 a 1000 GB (méně je lépe) a průměrnou rychlost (více je lépe) u nástrojů BitLocker a VeraCrypt ve srovnání šifrovacími algoritmy této aplikace. Pomocí nástrojů BitLocker a VeraCrypt byla data trvale dešifrována. Hodnoty u šifrovacích algoritmů této aplikace představují celkovou dobu trvání operace (čtení, dešifrování a zápis). Na Obrázku 6.5 je zobrazena závislost doby trvání dešifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací nástroje v režimu dešifrování						
BitLocker	**VeraCrypt**	**AES-128**	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Doba trvání pro 100, 500 a 1000 GB [h]						
~2,84	~3,60	~1,48	~1,53	~1,61	~3,01	~2,88
~14,22	~18,01	~7,41	~7,65	~8,06	~15,03	~14,42
~28,44	~36,03	~14,83	~15,29	~16,12	~30,06	~28,85
Průměrná rychlost [MB/s]						
~10	~8	~19	~19	~18	~9	~10

Tabulka 6.5: Srovnání šifrovacích nástrojů v režimu dešifrování.



Obrázek 6.5: Srovnání šifrovacích nástrojů v režimu dešifrování.

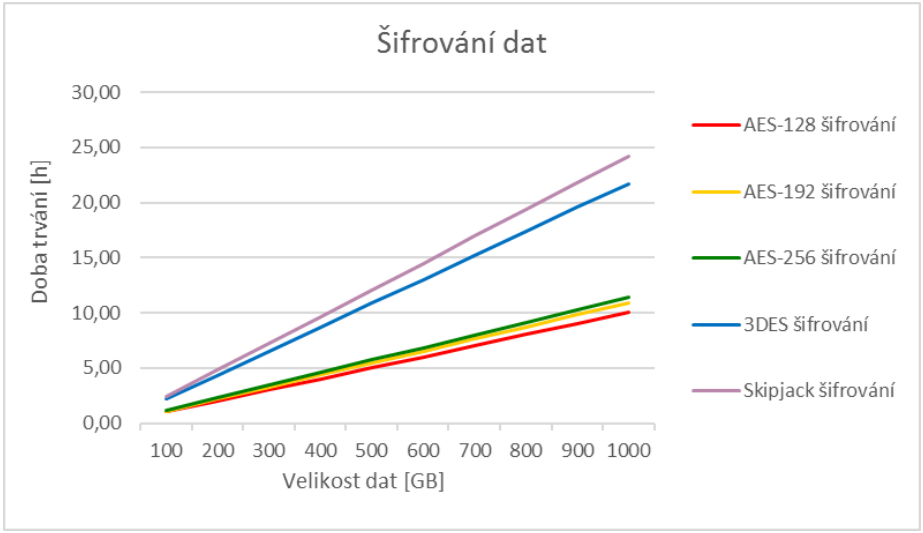
6.2 Průběh testování na druhé sestavě

Druhou testovací sestavou je tablet FUJITSU LIFEBOOK s procesorem Intel Core i5 5300U 2.20 GHZ, pamětí RAM 4 GB, 500 GB diskem SSHD a 64bitovým operačním systémem Windows 10 Education. Aplikací daná velikost bufferu dat je 82 252 800 B ~ 78 MB.

V Tabulce 6.6 je možné vidět následující výsledky měření šifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack: změřenou celkovou dobu trvání pro 1 GB (méně je lépe), dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.6 je zobrazena závislost doby trvání šifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací algoritmy v režimu šifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 1 GB [s]				
36,10	39,23	41,19	78,16	87,05
Doba trvání pro 100, 500 a 1000 GB [h]				
~1,00	~1,09	~1,14	~2,17	~2,42
~5,01	~5,45	~5,72	~10,86	~12,09
~10,03	~10,90	~11,44	~21,71	~24,18
Průměrná rychlost [MB/s]				
~28	~26	~25	~13	~12

Tabulka 6.6: Výsledky měření šifrování oddílů.

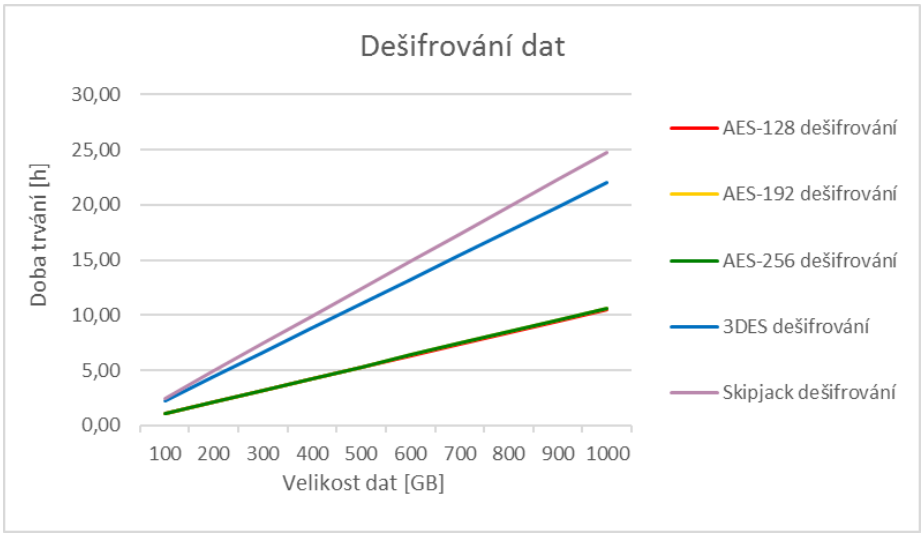


Obrázek 6.6: Závislost doby trvání šifrování dat na velikosti dat.

V Tabulce 6.7 je možné vidět následující výsledky měření dešifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack (pevný disk na toto měření nemá vliv): změřenou dobu trvání pro 1 GB (méně je lépe), dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.7 je zobrazena závislost doby trvání dešifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací algoritmy v režimu dešifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 1 GB [s]				
37,79	38,29	38,34	79,14	89,28
Doba trvání pro 100, 500 a 1000 GB [h]				
~1,05	~1,06	~1,06	~2,20	~2,48
~5,25	~5,32	~5,32	~10,99	~12,40
~10,50	~10,64	~10,65	~21,98	~24,80
Průměrná rychlost [MB/s]				
~27	~27	~27	~13	~11

Tabulka 6.7: Výsledky měření dešifrování oddílů.



Obrázek 6.7: Závislost doby trvání dešifrování dat na velikosti dat.

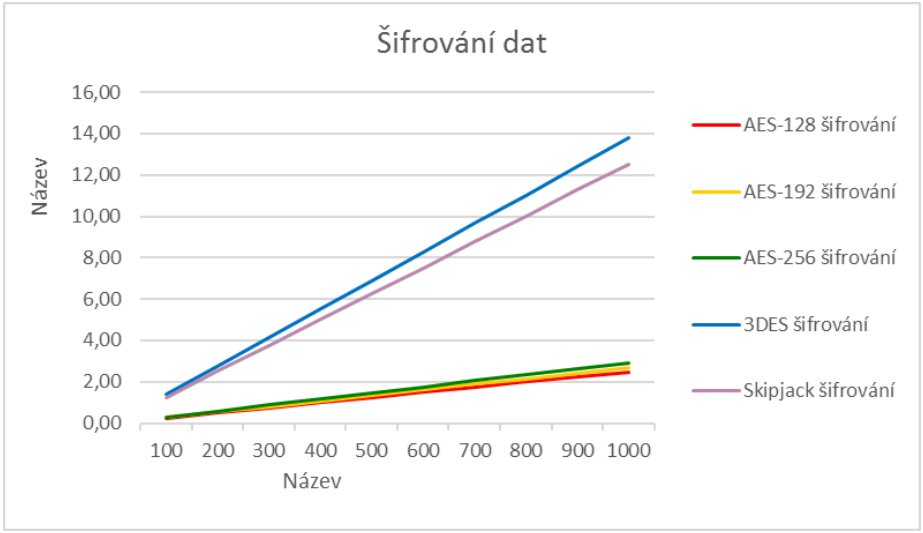
6.3 Průběh testování na třetí sestavě

Druhou testovací sestavou je notebook FUJITSU LIFEBOOK s procesorem Intel Core i7 4600M 2.90 GHZ, pamětí RAM 8 GB, 256 GB diskem SSD a 64bitovým operačním systémem Windows 8.1 Pro. Aplikací daná velikost bufferu dat je 82 252 800 B ~ 78 MB.

V Tabulce 6.8 je možné vidět následující výsledky měření šifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack: změřenou celkovou dobu trvání pro 1 GB (méně je lépe), dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.8 je zobrazena závislost doby trvání šifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací algoritmy v režimu šifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 1 GB [s]				
8,96	9,67	10,54	49,64	45,08
Doba trvání pro 100, 500 a 1000 GB [h]				
~ 0,25	~ 0,27	~ 0,29	~ 1,38	~ 1,25
~ 1,24	~ 1,34	~ 1,46	~ 6,89	~ 6,26
~ 2,49	~ 2,69	~ 2,93	~ 13,79	~ 12,52
Průměrná rychlost [MB/s]				
~ 114	~ 106	~ 97	~ 21	~ 23

Tabulka 6.8: Výsledky měření šifrování oddílů.

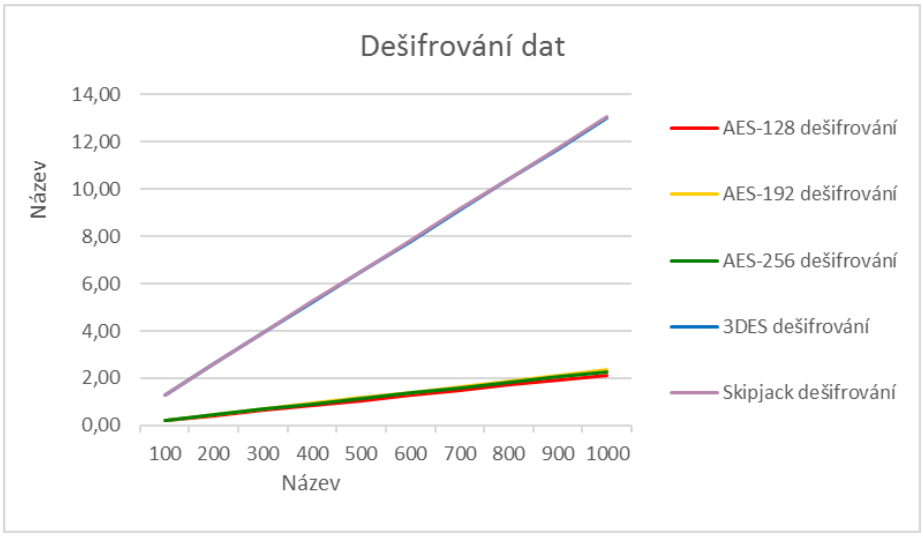


Obrázek 6.8: Závislost doby trvání šifrování dat na velikosti dat.

V Tabulce 6.9 je možné vidět následující výsledky měření dešifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack (pevný disk na toto měření nemá vliv): změřenou dobu trvání pro 1 GB (méně je lépe), dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.9 je zobrazena závislost doby trvání dešifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací algoritmy v režimu dešifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 1 GB [s]				
7,69	8,45	8,20	46,85	47,02
Doba trvání pro 100, 500 a 1000 GB [h]				
~0,21	~0,23	~0,23	~1,30	~1,31
~1,07	~1,17	~1,14	~6,51	~6,53
~2,14	~2,35	~2,28	~13,01	~13,06
Průměrná rychlost [MB/s]				
~133	~121	~125	~22	~22

Tabulka 6.9: Výsledky měření dešifrování oddílů.



Obrázek 6.9: Závislost doby trvání dešifrování dat na velikosti dat.

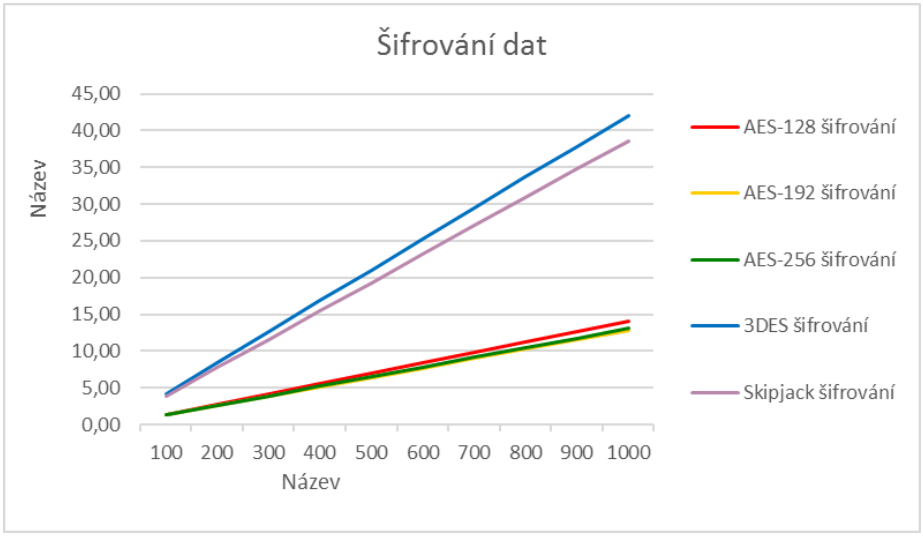
6.4 Průběh testování na čtvrté sestavě

Třetí testovací sestavou je notebook FUJITSU LIFEBOOK s procesorem Intel Core i5 4210M 2.60 GHZ, pamětí RAM 4 GB, 500 GB diskem SSHD a 64bitovým operačním systémem Windows 8.1 Pro. Aplikací daná velikost bufferu dat je 82 252 800 B ~ 78 MB.

V Tabulce 6.10 je možné vidět následující výsledky měření šifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack: změřenou celkovou dobu trvání pro 1 GB (méně je lépe), dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.10 je zobrazena závislost doby trvání šifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací algoritmy v režimu šifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 1 GB [s]				
50,37	46,12	46,94	151,36	138,98
Doba trvání pro 100, 500 a 1000 GB [h]				
~1,40	~1,28	~1,30	~4,20	~3,86
~7,00	~6,41	~6,52	~21,02	~19,30
~13,99	~12,81	~13,04	~42,05	~38,61
Průměrná rychlost [MB/s]				
~20	~22	~22	~7	~7

Tabulka 6.10: Výsledky měření šifrování oddílů.

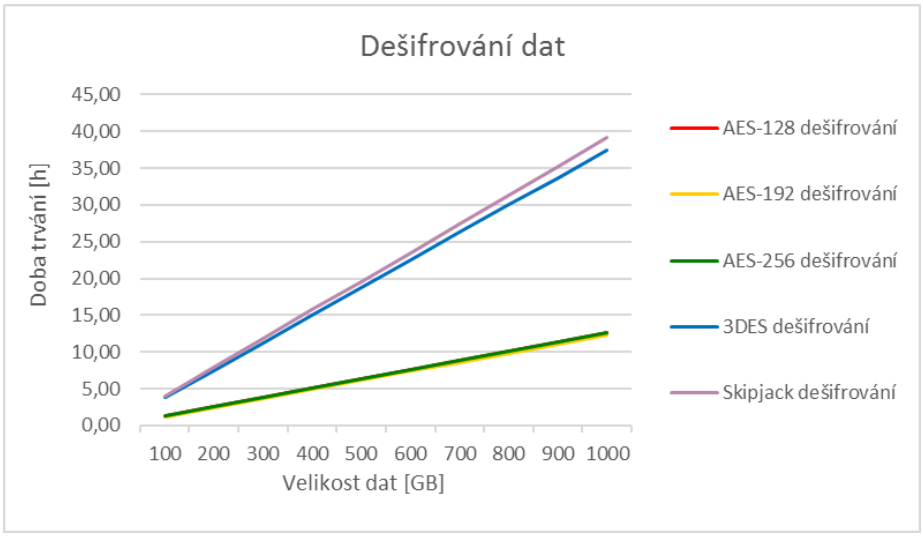


Obrázek 6.10: Závislost doby trvání šifrování dat na velikosti dat.

V Tabulce 6.11 je možné vidět následující výsledky měření dešifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack (pevný disk na toto měření nemá vliv): změřenou dobu trvání pro 1 GB (méně je lépe), dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.11 je zobrazena závislost doby trvání dešifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací algoritmy v režimu dešifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 1 GB [s]				
45,18	44,29	45,52	135,00	141,23
Doba trvání pro 100, 500 a 1000 GB [h]				
~1,25	~1,23	~1,26	~3,75	~3,92
~6,27	~6,15	~6,32	~18,75	~19,61
~12,55	~12,30	~12,64	~37,50	~39,23
Průměrná rychlost [MB/s]				
~23	~23	~22	~8	~7

Tabulka 6.11: Výsledky měření dešifrování oddílů.



Obrázek 6.11: Závislost doby trvání dešifrování dat na velikosti dat.

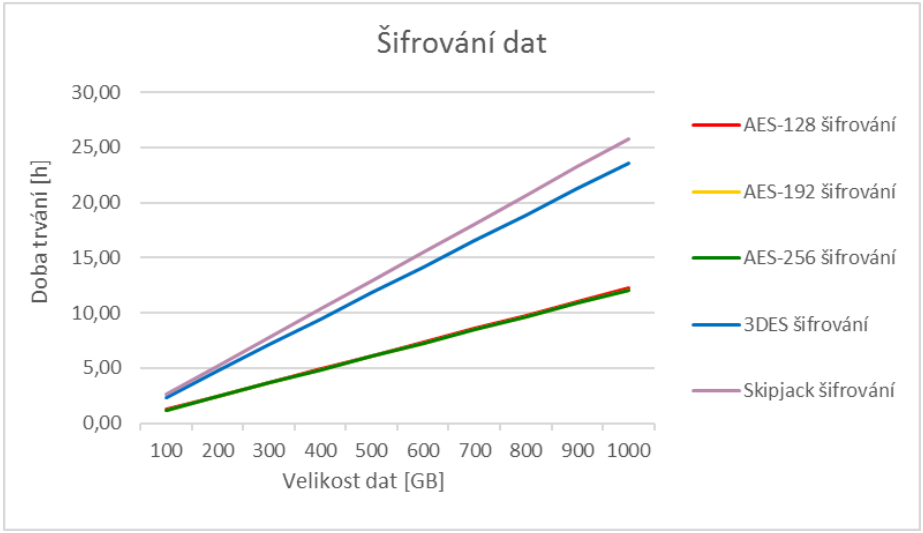
6.5 Průběh testování na páté sestavě

Pátou testovací sestavou je ultrabook FUJITSU LIFEBOOK s procesorem Intel Core i5 5200U 2.20 GHZ, pamětí RAM 4 GB, 500 GB diskem SSHD a 64bitovým operačním systémem Windows 7 Professional. Aplikací daná velikost bufferu dat je 82 252 800 B ~ 78 MB.

V Tabulce 6.12 je možné vidět následující výsledky měření šifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack: změřenou celkovou dobu trvání pro 1 GB (méně je lépe), dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.12 je zobrazena závislost doby trvání šifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací algoritmy v režimu šifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 1 GB [s]				
43,95	43,50	43,43	84,99	92,89
Doba trvání pro 100, 500 a 1000 GB [h]				
~1,22	~1,21	~1,21	~2,36	~2,58
~6,10	~6,04	~6,03	~11,80	~12,90
~12,21	~12,08	~12,06	~23,61	~25,80
Průměrná rychlost [MB/s]				
~23	~24	~24	~12	~11

Tabulka 6.12: Výsledky měření šifrování oddílů.

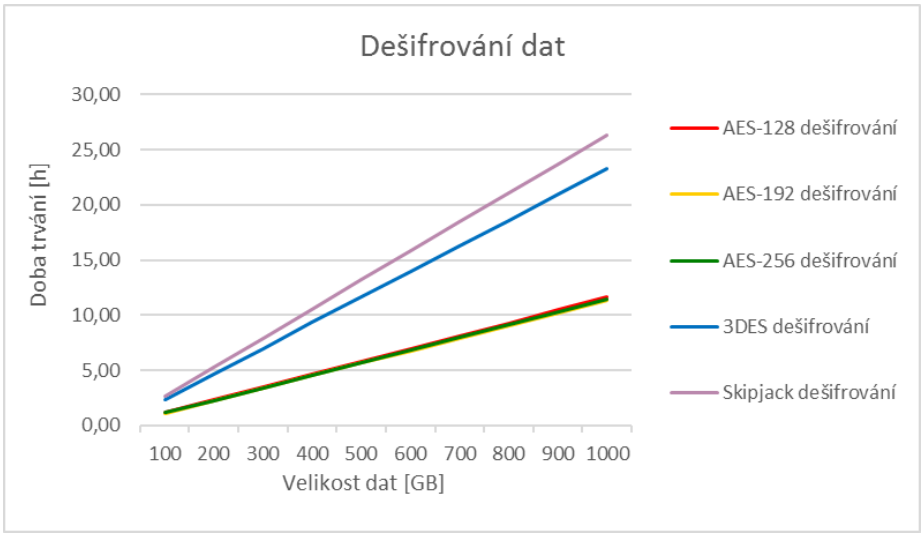


Obrázek 6.12: Závislost doby trvání šifrování dat na velikosti dat.

V Tabulce 6.13 je možné vidět následující výsledky měření dešifrování dat pro jednotlivé šifrovací algoritmy AES-128, AES-192, AES-256, 3DES a Skipjack (pevný disk na toto měření nemá vliv): změřenou dobu trvání pro 1 GB (méně je lépe), dobu trvání pro 100, 500 a 1000 GB a průměrnou rychlost (více je lépe). Na Obrázku 6.13 je zobrazena závislost doby trvání dešifrování dat na velikosti dat od 100 GB do 1 TB.

Šifrovací algoritmy v režimu dešifrování				
AES-128	**AES-192**	**AES-256**	**3DES**	**Skipjack**
Změřená doba trvání pro 1 GB [s]				
41,84	40,67	41,13	83,78	94,92
Doba trvání pro 100 GB [h]				
~1,16	~1,13	~1,14	~2,33	~2,64
Doba trvání pro 500 GB [h]				
~5,81	~5,65	~5,71	~11,64	~13,18
Doba trvání pro 1000 GB [h]				
~11,62	~11,30	~11,43	~23,27	~26,37
Průměrná rychlost [MB/s]				
~24	~25	~25	~12	~11

Tabulka 6.13: Výsledky měření dešifrování oddílů.



Obrázek 6.13: Závislost doby trvání dešifrování dat na velikosti dat.

6.6 Diskuze výsledků

Z údajů měření na první počítačové sestavě vyplývá, že samotné čtení a zápis dat z pevného disku se zvoleným oddílem trvá nezanedbatelnou dobu a v případě stovek GB se může jednat i o celé hodiny. Samotný proces šifrování a dešifrování se u jednotlivých šifrovacích algoritmů také výrazně liší a lze vidět téměř pětinásobný rozdíl mezi vysoce optimalizovaným algoritmem AES a algoritmy 3DES a Skipjack v režimu šifrování. Na druhou stranu testy ukázaly, že zatímco u algoritmů 3DES a Skipjack se doba trvání operace dešifrování příliš neliší od doby trvání operace šifrování, u algoritmu AES je dešifrování téměř dvakrát pomalejší.

V průběhu testování na první sestavě také proběhlo srovnání dvou vybraných v současnosti udržovaných šifrovacích nástrojů BitLocker a VeraCrypt s implementovanými šifrovacími algoritmy v této aplikaci. Zatímco algoritmy 3DES a Skipjack podávaly v režimu šifrování i dešifrování podobné hodnoty doby trvání operace šifrování i dešifrování jako BitLocker a VeraCrypt, algoritmus AES je ve všech třech verzích nesrovnatelně rychlejší a i součet obou hodnot doby trvání operace šifrování a dešifrování se pohybuje kolem jedné operace ostatních nástrojů. Na druhou stranu umožňují ostatní nástroje sofistikované šifrování On-the-fly, které umožňuje přístup k datům bez potřeby je všechna dešifrovat, nástroje jsou komplexní a umějí šifrovat i systémový oddíl včetně zavedené pre-boot autentizace po spuštění počítače.

Z celkového srovnání testování mezi jednotlivými sestavami lze vyvodit, že výsledný rozdíl mezi šifrováním a dešifrováním je poměrně malý, zatímco rozdíl mezi použitím kryprografického standardu AES a standardy 3DES a Skipjack je naopak markantní. Dále byla ověřena podpora platforem Windows 7 výše.

Kapitola 7

Závěr

Cílem této práce bylo prostudovat literaturu týkající se šifrování disků se zaměřením na algoritmus AES, navrhnout a implementovat aplikační řešení pro platformu Windows, otestovat jej a zhodnotit výsledky.

Byly prostudovány a popsány oblasti související s šifrováním dat, jmenovitě symetrická kryptografie, nejznámější současné blokové šifrovací algoritmy, blokové módy operací nad těmito algoritmy, metody šifrování disku a také byly srovnány existující šifrovací nástroje.

Poté bylo navrženo vícejazyčné aplikační řešení, nicméně při implementaci bylo v důsledku náročnosti a nedostatku času ustoupeno od původního záměru umožnit uživateli šifrovat systémový disk se zavedením pre-boot autentizace. Také byl změněn vlastní návrh blokového módu vycházejícího z XTS za mód CBC a zarovnávání bloků pomocí jednoho bitu 1 a potřebného počtu bitů 0 za CTS z důvodu velmi úzké provázanosti blokových módů s šifrovacími algoritmy v knihovně a také jejich optimalizované implementaci. Aplikace je umožňuje šifrovat a dešifrovat nepoužívaný oddíl po zadání vstupního hesla, které se spolu s klíče ukládá v šifrované podobě na disku.

Aplikace byla implementována s pomocí knihovny Crypto++ a rozhraní Windows API. Lze šifrovat i dešifrovat nepoužívané oddíly pomocí šifrovacích algoritmů AES-128, AES-192, AES-256, 3DES a Skipjack v blokovém módu CBC-CTS, kde je možné buď zadat heslo, nebo nahrát exportovaný klíč. Více informací je uvedeno v příloze manuál.

Byly otestovány a diskutovány výsledky měření vlastností aplikace z hlediska funkčnosti a rychlosti operací v závislosti na velikosti dat.

V budoucnu je možné dále doplnit aplikaci o pre-boot autentizaci, šifrování systémového oddílu, doplnění samoopravných kódů, vylepšit uživatelské rozhraní, a optimalizovat kód.

Literatura

[1] Alam, A. *Disk Encryption* Norwegian University of Science and Technology, 2009. Vedoucí práce Danilo Gligoroski.

[2] Bellovin, S. : *Frank Miller: Inventor of the One-Time Pad. Cryptologia*, roč. 35, č. 3, 2011: s 203 – 222.

[3] Cox, B.; Forget, A.; Horrocs, C.; aj. *CipherShed: Secure Encryption Software* [online]. 2016 Dostupné z: <https://www.ciphershed.org/>.

[4] Daemen, J.; Rijmen, V. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002. 238 s. ISBN 3-540-42580-2.

[5] Delfs, H.; Knebl, H. *Introduction to Cryptography: Principles and Applications*. 3. Springer Berlin Heidelberg, 2015. ISBN 978-3-662-47974-2.

[6] DiskCryptor. *DiskCryptor wiki* [online]. 2012 Dostupné z: <https://diskcryptor.net/wiki/Main_Page>.

[7] El-Fotouh, M.; Diepold, K. : *A New Narrow Block Mode of Operations for Disk Encryption*. 2008.

[8] ForensicsWiki. *PGPDisk* [online]. 2008 Dostupné z: <http://www.forensicswiki.org/wiki/PGPDisk>.

[9] ForensicsWiki. *DiskCryptor* [online]. 2009 Dostupné z: <http://www.forensicswiki.org/wiki/DiskCryptor>.

[10] ForensicsWiki. *FreeOTFE* [online]. 2009 Dostupné z: <http://www.forensicswiki.org/wiki/FreeOTFE>.

[11] ForensicsWiki. *TrueCrypt* [online]. 2014 Dostupné z: <http://www.forensicswiki.org/wiki/TrueCrypt>.

[12] ForensicsWiki. *BitLocker Disk Encryption* [online]. 2015 Dostupné z: <http://www.forensicswiki.org/wiki/BitLocker_Disk_Encryption>.

[13] Halevi, S.; Rogaway, P. : *A Parallelizable Enciphering Mode*. 2003.

[14] Halevi, S.; Rogaway, P. : *A Tweakable Enciphering Mode*. 2003.

[15] Idrassi, M. *Security Requirements and Precations* [online]. 2014 Dostupné z: <https://veracrypt.codeplex.com/wikipage?title=Security%20Requirements%20and%20Precautions>.

[16] IDRIX. *VeraCrypt* [online]. 2016 Dostupné z: <https://veracrypt.codeplex.com/>.

[17] Liskov, M.; Rivest, R.; Wagner, D. : *Tweakable Block Ciphers*. 2002.

[18] Macek, R. : *Bezpečnostní aspekty souborových systémů a bezpečnost dat zajištěná šifrováním disků*. Bakalářská práce, Masarykova univerzita, Fakulta informatiky, Brno, 2012.

[19] McGrew, D.; Fluhrer, S. : *The Extended Codebook (XCB) Mode of Operation*. 2004.

[20] Menezes, A.; van Oorschot, P.; Vanstone, S. *Handbook of Applied Cryptography*. CRC Press, 1996. ISBN 0-8493-8523-7.

[21] Microsoft. *BitLocker Drive Encryption* [online]. 2010 Dostupné z: <https://technet.microsoft.com/cs-cz/library/cc725719(v=ws.10).aspx>.

[22] NIST : *FIPS PUB 81: DES Modes of Operation*. 1980.

[23] NIST : *FIPS PUB 185: Escrowed Encryption Standard*. 1994.

[24] NIST. *First AES Candidate Conference* [online]. 1998 Dostupné z: <http://www.ieee-security.org/Cipher/ConfReports/conf-rep-aes.html>.

[25] NIST : *Skipjack and KEA Algorithm Specification*. 1998.

[26] NIST : *FIPS PUB 46-3: DES*. 1999.

[27] NIST. *BLOCK CIPHERS: Approved Algorithms* [online]. 2000 Dostupné z: <http://csrc.nist.gov/groups/ST/toolkit/block_ciphers.html>.

[28] NIST : *FIPS PUB 197: AES*. 2001.

[29] NIST : *Special PUB 800-38A: Recommendation for Block Cipher Modes of Operation*. 2001.

[30] NIST : *Clarification to the Skipjack Algorithm Specification*. 2002.

[31] NIST : *IEEE Std 1619-2007: IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices*. 2008.

[32] NIST : *Special PUB 800-38E: Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*. 2010.

[33] NIST : *Special PUB 800-67 Rev 1: TDEA*. 2012.

[34] iSEC Partners : *Open Crypto Audit Project TrueCrypt*. 2016.

[35] Rogaway, P. : *Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC*. 2004.

[36] Symantec. *PGP Whole Disk Encryption* [online]. 2010 Dostupné z: <https://www.symantec.com/content/cs/cz/enterprise/fact_sheets/b-PGP_Whole_Disk_Encryption_CZ.pdf>.

[37] Techworld. *Bitlocker review* [online]. 2010 Dostupné z: <http://www.techworld.com/review/encryption/bitlocker-review-3212400/>.

[38] Techworld. *PGP Whole Disk Encryption review* [online]. 2010 Dostupné z: <http://www.techworld.com/review/encryption/pgp-whole-disk-encryption-review-3212415/>.

[39] Techworld. *TrueCrypt review* [online]. 2010 Dostupné z: <http://www.techworld.com/review/encryption/truecrypt-review-3212403/>.

[40] Techworld. *FreeOTFE review* [online]. 2011 Dostupné z: <http://www.techworld.com/review/encryption/freeotfe-review-3264498/>.

[41] Techworld. *DiskCryptor review* [online]. 2012 Dostupné z: <http://www.techworld.com/review/security-software/diskcryptor-review-3408069/>.

[42] Zeghid, M.; Machhout, M.; Khriji, L.; aj. : *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, roč. 1, č. 3, 2007: s 745 – 750, ISSN 1307 - 6892.

Přílohy

Seznam příloh

Příloha A

Obsah CD

Datové médium obsahuje elektronickou verzi technické zprávy, manuálů a související zdrojové kódy.

A.1 Struktura CD

- `./projekt.pdf` : písemná zpráva
- `./bin/` : adresář se spustitelnou aplikací
- `./latex/` : zdrojový tvar písemné zprávy
- `./manual/` : návody k použití aplikace
- `./DiskEncryption/` : zdrojové kódy včetně zkompilované knihovny Crypto++ verze 5.6.3

Příloha B

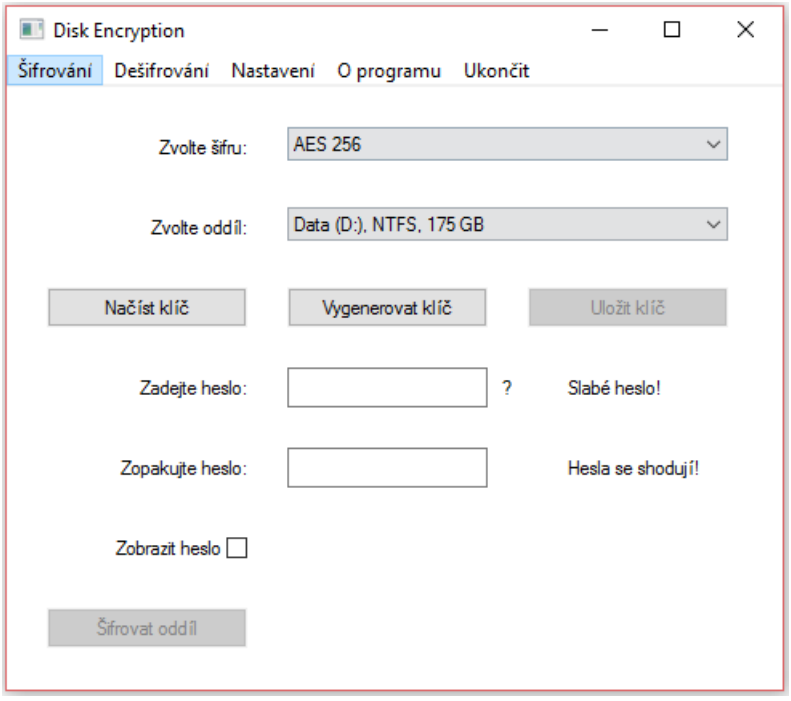
Český manuál

B.1 Požadavky

Aplikace je určena především pro operační systém Windows 8.1 a Windows 10. Pro své spuštění vyžaduje administrátorská práva a existenci adresáře zvaného `locales` s alespoň jedním jazykovým souborem typu JSON.

B.2 Spuštění aplikace

Naběhnutí aplikace může trvat až několik vteřin, pokud její start selže, pravděpodobně se jedná o chybu související s načítáním jazykových souborů. Na Obrázku B.1 lze vidět hlavní okno aplikace po spuštění.

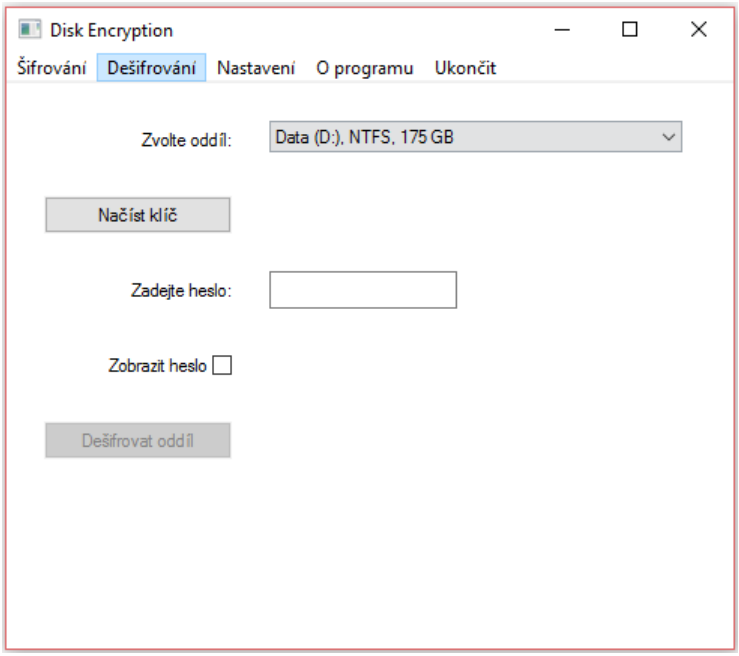


Obrázek B.1: Hlavní okno aplikace po spuštění.

Po spuštění se objeví hlavní okno aplikace s těmito ovládacími prvky (odshora): **Lišta s menu** - umožňuje se přepnout mezi šifrováním a dešifrováním, nastavit jazyk aplikace, zobrazit krátké info o aplikace nebo ji ukončit. **Seznam diskových oddílů** - u každé položky v seznamu je uvedené (pokud bylo možné tuto informaci získat) uživatelské jméno oddílu, diskovou jednotku, souborový systém a velikost oddílu. **Seznam šifrovacích algoritmů** - aplikace podporuje následující algoritmy: AES-128, AES-192, AES-256, 3DES a Skipjack. V případě výběru nové položky je odebrán klíč a je nutné jej importovat nebo vygenerovat znovu. Importovat lze libovolný podporovaný klíč bez ohledu na zvolenou položku seznamu. **Tlačítka pro importování, generování a exportování klíče** - importování klíče vyžaduje formát specifický pro tuto aplikaci (ideálně jeden z exportovaných), ale tato možnost se nedoporučuje! Uživatel by měl vygenerovat nový klíč a exportovat jej jako zálohu. **První vstup pro heslo s ověřením jeho síly** - v průběhu opakování hesla se vedle vstupu zobrazuje stav jeho síly. Požadavky na minimální tvar lze zobrazit přejetím myši přes otazník vedle vstupu. **Druhý vstup pro heslo** - v průběhu zadávání hesla se vedle vstupu zobrazuje informace, zda obě hesla souhlasí. **Přepínač zobrazení hesel bez masky** - je možné přepnout zobrazení hesel mezi maskou a otevřeným tvarem. **Tlačítko pro spuštění šifrování** - toto tlačítko bude dostupné, až bude aplikace mít šifrovací klíč, hesla budou dostatečně bezpečná a budou souhlasit.

B.3 Okno dešifrování

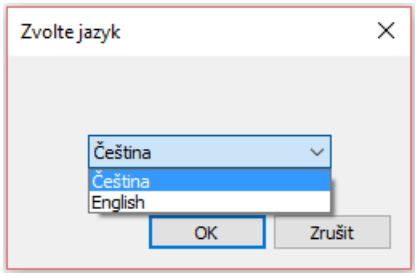
Na Obrázku B.2 lze vidět okno pro dešifrování. Přepnutím na dešifrování se objeví okno s ovládacími prvky podobnými oknu pro šifrování bez seznamu šifrovacích algoritmů, tlačítek pro generování a exportování klíče a druhého vstupu pro heslo. Tlačítko pro spuštění dešifrování bude dostupné, až bude aplikace mít šifrovací klíč nebo začne uživatel zadávat heslo. Importovaný klíč má přednost před heslem.



Obrázek B.2: Okno pro dešifrování.

B.4 Dialog nastavení

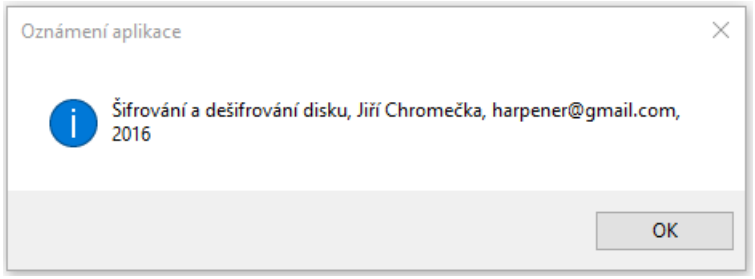
Na Obrázku B.3 lze vidět dialog s nastavením jazyka.



Obrázek B.3: Dialog s nastavením jazyka.

B.5 Informace o aplikaci

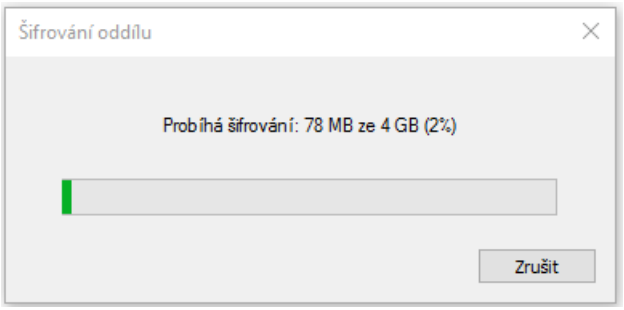
Na Obrázku B.4 lze vidět krátké info o aplikaci včetně kontaktu na autora.



Obrázek B.4: Krátké info o aplikaci.

B.6 Kroky šifrování

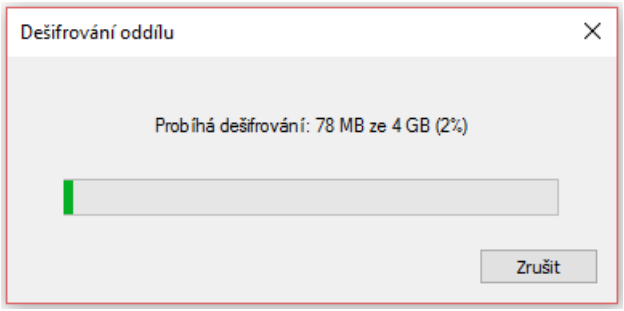
Po stisknutí tlačítka šifrování dojde ke kontrole parametrů, v adresáři s aplikací se vytvoří soubor zvaný `encrypted_O.BIN` (O je jednotka s oddílem) a v tomto souboru budou uloženy potřebná data pro ověření kontrolního hesla a dešifrování oddílu. Bez tohoto souboru aplikace nebude považovat oddíl za šifrovaný! Pokud takový soubor již existuje (oddíl je již šifrovaný), aplikace neumožní pokračovat dále. Poté aplikace zjistí informace o zvoleném oddílu a pokusí se jej uzamknout. Musí se jednat o nepoužívaný oddíl! Pokud se objeví chybová hláška, zastavte jakékoli související běžící aplikace a služby, a poté restartujte aplikaci. Pokud nedojde k žádným chybám, objeví se dialog s průběhem operace šifrování, viditelný na Obrázku B.5. V tomto okamžiku jsou data šifrována a jakékoli vyrušení (např. vypnutí systému) může znamenat jejich ztrátu! Operaci lze v průběhu kdykoli vrátit zpět stisknutím tlačítka Zrušit, kdy se spustí dešifrování dat, které už nelze přerušit. Po úspěšném skončení operace se aplikace automaticky ukončí.



Obrázek B.5: Průběh šifrování.

B.7 Kroky dešifrování

Po stisknutí tlačítka dešifrování dojde ke kontrole parametrů a v případě, že byl importován klíč, se přeskočí ověřování hesla. Jinak je zkontrolována existence souboru zvaného `encrypted_O.BIN`, kde O je jednotka s oddílem, a ověřena správnost kontrolního hesla. Další průběh je podobný šifrování, a pokud nedojde k žádným chybám, objeví se dialog s průběhem operace dešifrování, viditelný na Obrázku B.6. V tomto okamžiku jsou data dešifrována a jakékoli narušení typu výpadek proudu může znamenat jejich ztrátu! Operaci lze v průběhu kdykoli vrátit zpět stisknutím tlačítka Zrušit, kdy se spustí šifrování dat, které už nelze přerušit. Po úspěšném skončení operace se aplikace automaticky ukončí.



Obrázek B.6: Průběh dešifrování.

B.8 Přidání nového jazyka

Pro přidání nového jazyka postačí vytvořit kopii jednoho z jazykových souborů typu JSON v adresáři zvaném `locales` a upravit jej následujícím způsobem. Na každém řádku ve tvaru `"A": "B",`, text `A` musí zůstat stejný, text `B` lze přeložit do zvoleného jazyka, a na prvním řádku ve tvaru `"name": "J",` přijde text J nastavit na název zvoleného jazyka.

Appendix C

English manual

C.1 Requirements

The application is intended for use mainly on the operation systems Windows 8.1 and Window 10. It requires the administration rights and the existence of a directory named `locales` with at least one language file of the JSON type.

C.2 Application launch

The application launch can last several seconds and if it fails, the probable cause is related to the loading of the language files. See Figure C.1 for the main application window after its launch.

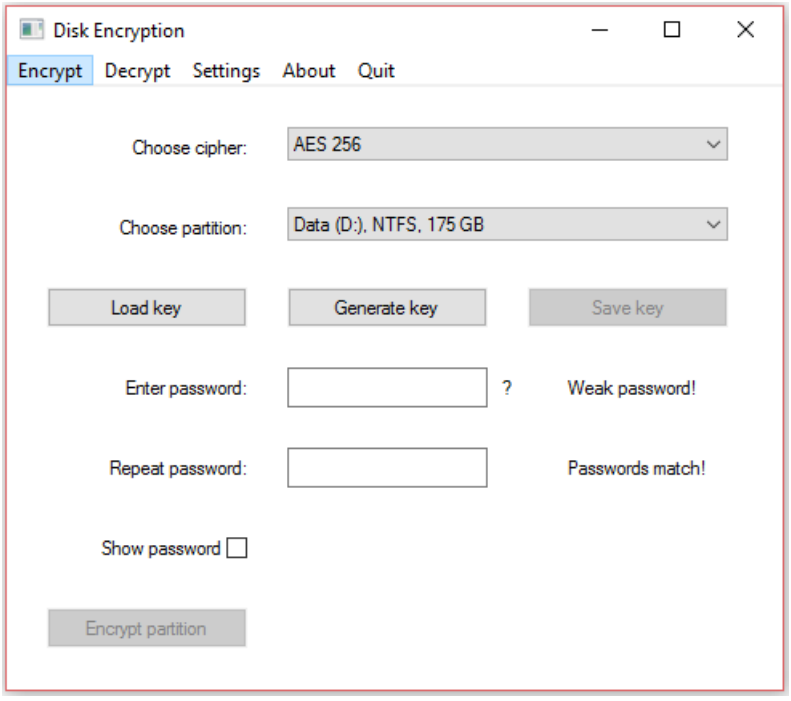


Figure C.1: Main application window after its launch.

After launch there will be a main application window with the following control items (from top): **Menu bar** - allows switching between the encryption and the decryption, to set the application language, to show a short application info or to quit the application. **List of partitions** - each item of the list consists of (if the information was available) a volume label, a drive letter, a file system and a partition size. **List of the encryption algorithms** - the application supports the following algorithms: the AES-128, the AES-192, the AES-256, the 3DES and the Skipjack. In case of selecting a new item, the key is removed and it is necessary to import or generate a new one. Any supported key can be imported regardless to the selected encryption algorithm. **Buttons for importing, generating and exporting a key** - importing a key requires a certain format specific for this application (ideally an exported one), but this option is not recommended! The user should generate a new key and export it as a backup. **First password input with its strength status** - while the user enters password, its strength sufficiency is shown next to the input. Requirements for its minimal sufficiency can be shown by moving mouse over the question mark next to the input. **Second password input** - while the user repeats the password, a match status is shown next to the input. **Show password switch** - it is possible to switch between a password mask and a clear text. **Button for the encryption start** - this button will be enabled once the application has an encryption key, the passwords have a sufficient strength and are matching.

C.3 Decryption window

See Figure C.2 for the decryption window. By switching to the decryption window, the control items are shown similar to the ones in the encryption window without the list of the encryption algorithms, the buttons for generating and exporting the key and the second password input. Button for the decryption start will be enabled once the application has an encryption key or the user starts entering a password. Imported key has a priority over the entered password.

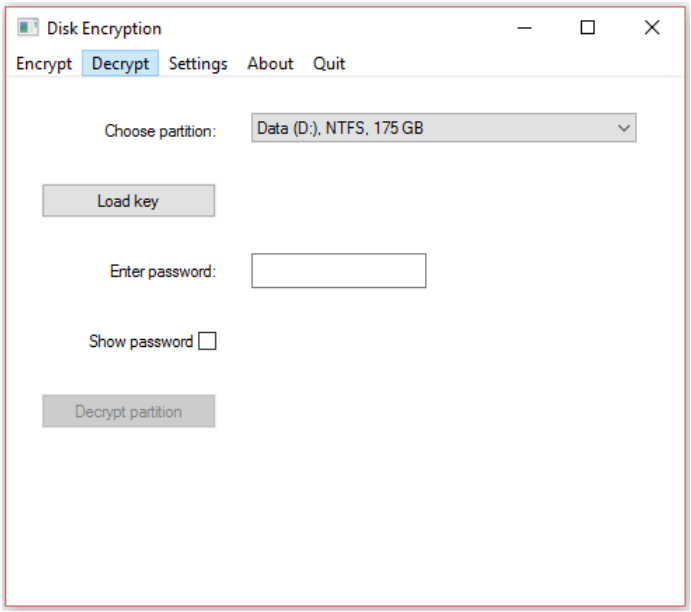


Figure C.2: Decryption window.

C.4 Settings dialog

See Figure C.3 for a dialog used for selecting a language.

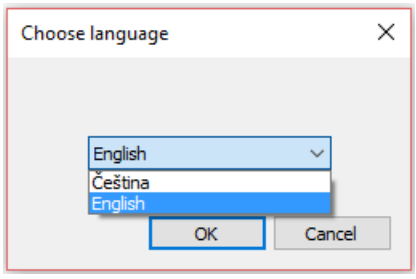


Figure C.3: A dialog used for selecting a language.

C.5 Information about the application

See Figure C.4 for a short info about the application including author contact.

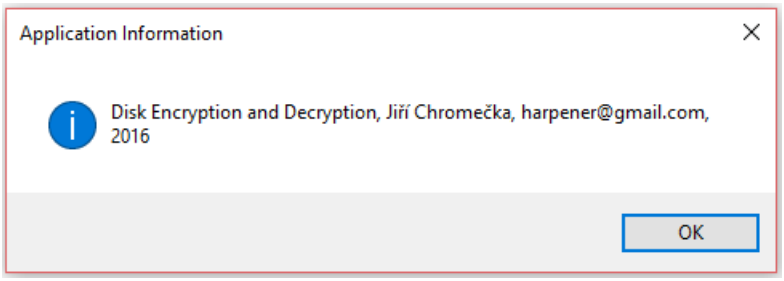


Figure C.4: Short info about the application.

C.6 Encryption steps

After the encryption button is pressed, the encryption parameters are checked, a file named `encrypted_D.BIN` (D stands for the drive letter) will be created in the same directory as the one with the application and it will contain the necessary data for the verifying of the password and the decryption of the partition. Without this file the application will recognize the partition as encrypted! If such a file already exists (the partition is already encrypted), the application will not allow continuing. After that application retrieves the additional partition information and attempts to lock the partition. It has to be an unused partition! If an error message appears, stop any related running applications or services and then restart the application. Upon a success there will appear a dialog with the progress of the encryption operation, see Figure C.5. From this moment the data are being encrypted and any disruption (e.g. a system shutdown) will lead to their loss! The operation in progress can be reverted at any time by pressing the Cancel button and a data encryption will start. Upon a successful finish the application will automatically quit.

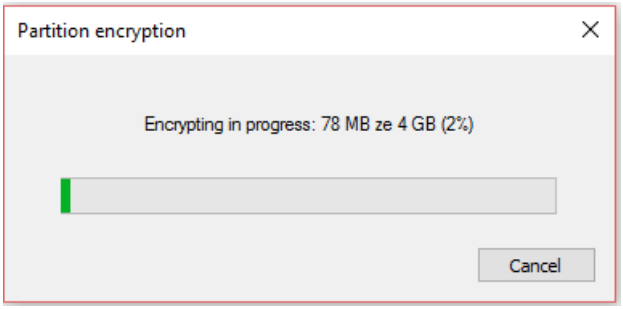


Figure C.5: Encryption progress.

C.7 Decryption steps

After the decryption button is pressed, the decryption parameters are checked and in case of importing the key, the password verification will be skipped. Otherwise, the existence of a file named `encrypted_D.BIN` (D stands for the drive letter) will be checked and the password will be verified. Following steps are similar to the encryption ones and upon a success there will appear a dialog with the progress of the decryption operation, see Figure C.6. From this moment the data are being decrypted and any disruption (e.g. a system shutdown) will lead to their loss! The operation in progress can be reverted at any time by pressing the Cancel button and a data decryption will start. Upon a successful finish the application will automatically quit.

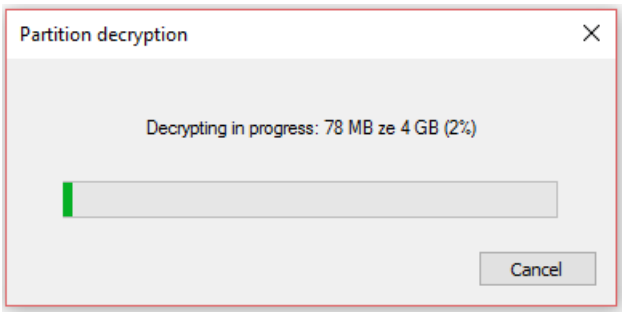


Figure C.6: Decryption progress.

C.8 New language addition

To add a new language, create a copy of one of the language files of the JSON type in the directory named `locales` and edit it the following way. For each row matching the pattern `"A": "B",`, the text `A` has to remain the same and the text `B` can be translated into selected language and for the first row matching the pattern `"name": "L",`, the text J is to be set to the selected language name.