VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ÚTOK NA ŠIFROVANÉ DISKOVÉ ODDÍLY S VYUŽITÍM GPU

ATTACK ON ENCRYPTED DISK VOLUMES USING GPU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ONDŘEJ SEDLO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK HRANICKÝ**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19362/2016/xsedlo02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Sedlo Ondřej**
Obor: Informační technologie
Téma: **Útok na šifrované diskové oddíly s využitím GPU**
Attack on Encrypted Disk Volumes Using GPU
Kategorie: Bezpečnost

Pokyny:

1. Seznamte se s nástrojem Fitcrack/Wrathion.
2. Nastudujte techniky šifrování diskových oddílů nástroji jako TrueCrypt, VeraCrypt, FileVault, apod.
3. Navrhněte rozšíření nástroje z bodu 1, které umožní útok na šifrované diskové oddíly za účelem nalezení správného hesla.
4. Navržené rozšíření implementujte (včetně kódu pro akceleraci pomocí GPU).
5. Experimentálně ověřte efektivitu a použitelnost implementovaného řešení.
6. Zhodnoťte dosažené výsledky.

Literatura:

- ZANG, L. ZHOU, Y., J. Fan. The forensic analysis of encrypted Truecrypt volumes. *In Proceedings of 2014 International Conference Progress in Informatics and Computing (PIC)*. Shanghai: IEEE 2014. s. 405-409. ISBN 978-1-4799-2030-3.
- MENEZES, Alfred J., VAN OORSCHOT, Paul C., VANSTONE, Scott A. Handbook of Applied Cryptography. Boca Raton (Florida): CRC Press 1999. 810 s. ISBN 978-8189836122.
- A další dle dohody s vedoucím.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% 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: **Hranický Radek, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

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

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této práce je vytvořit rozšíření nástroje Fitcrack, které bude obnovovat hesla diskových oddílů šifrovaných nástrojem TrueCrypt. V úvodní části je popsán nástroj Fitcrack, který slouží k obnově hesel různých formátů souborů a framework OpenCL, který využívá k akceleraci výpočtů. Dále se práce zabývá nástrojem TrueCrypt, jeho možnostmi použití, analýzou formátu šifrovaných diskových oddílů a kryptografickými algoritmy, které TrueCrypt k šifrování diskových oddílů používá. Dále je v práci uveden návrh a implementace modulu Fitcracku k obnově hesel diskových oddílů šifrovaných TrueCryptem. Na konci práce jsou uvedeny výsledky experimentů srovnání rychlosti CPU a GPU.

Abstract

The aim of this thesis is to create an extension of the Fitcrack tool, which will recover passwords from disk volumes encrypted by TrueCrypt. The introduction part describes the Fitcrack tool which is used for password recovery and supports many different formats of files, and OpenCL framework which is used for computation acceleration. The thesis also deals with the TrueCrypt tool, and its possibilities of use, encrypted disk volumes analysis and cryptographic algorithms used by TrueCrypt to encrypt disk volumes. Furthermore, the work includes a concept and implementation of the Fitcrack module which will be used to recover passwords from disk volumes encrypted by TrueCrypt. At the end of the work we can see results of our experiments that compare speeds of CPU and GPU.

Klíčová slova

Fitcrack, TrueCrypt, PBKDF2, SHA-512, RIPEMD-160, Whirlpool, AES, Serpent, Twofish, XTS

Keywords

Fitcrack, TrueCrypt, PBKDF2, SHA-512, RIPEMD-160, Whirlpool, AES, Serpent, Twofish, XTS

Citace

SEDLO, Ondřej. *Útok na šifrované diskové oddíly s využitím GPU*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Hranický

Útok na šifrované diskové oddíly s využitím GPU

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením inženýra Radka Hranického. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ondřej Sedlo
17. května 2017

Poděkování

Chtěl bych poděkovat inženýrovi Radkovi Hranickému za cenné rady a odbornou pomoc při psaní bakalářské práce.

Obsah

1 Úvod **3**

2 Fitcrack **4**
2.1 Generátory hesel . . . 4
2.2 Obnova hesel . . . 5
2.2.1 CPU . . . 5
2.2.2 GPU . . . 5
2.3 Modularita . . . 6
2.4 OpenCL . . . 8
2.4.1 Model platformy . . . 8
2.4.2 Výpočetní model . . . 8
2.4.3 Paměťový model . . . 8
2.4.4 Programovací model . . . 9

3 Kryptografické algoritmy **10**
3.1 Generátor náhodných čísel . . . 10
3.2 PBKDF2 . . . 10
3.3 Hešování . . . 11
3.3.1 SHA-512 . . . 12
3.3.2 RIPEMD-160 . . . 12
3.3.3 Whirlpool . . . 13
3.3.4 CRC32 . . . 15
3.4 Šifrovací algoritmy . . . 15
3.4.1 AES . . . 15
3.4.2 Serpent . . . 17
3.4.3 Twofish . . . 18
3.4.4 Konkatenace algoritmů . . . 18
3.5 Režimy blokových šifer . . . 18
3.5.1 XTS . . . 18

4 TrueCrypt **20**
4.1 Šifrovaný svazek . . . 20
4.1.1 Skrytý svazek . . . 20
4.2 Šifrovaný operační systém . . . 21
4.2.1 Skrytý operační systém . . . 21
4.3 Hlavička . . . 22
4.4 Datová oblast . . . 22
4.5 Keyfile . . . 24

4.6 Ostatní nástroje k šifrování diskových oddílů 24

5 Návrh modulu **25**

6 Implementace **27**
6.1 FitXtractor 27
6.2 Fitcrack 27

7 Experimenty **29**
7.1 Všechny algoritmy 29
7.2 Hešovací algoritmy 30
7.3 Šifrovací algoritmy 30
7.4 Shrnutí výsledků 31

8 Závěr **32**

Literatura **33**

Přílohy **35**

A Obsah přiloženého paměťového média **36**

Kapitola 1

Úvod

Šifrování přeměňuje data do podoby, ve které je utajen jejich původní smysl. K dešifrování je nutné znát klíč a daný kryptografický algoritmus. Šifrování se používá k zaručení důvěrnosti a integrity dat. V současnosti existuje mnoho nástrojů, které mohou provádět šifrování diskových oddílů. Jedním z nejznámějších nástrojů k vytváření a údržbě šifrovaných diskových oddílů je TrueCrypt. Je to dáno především tím, že používá mnoho šifrovacích algoritmů a jejich konkatenace, čímž se stává jedním z nejlépe zabezpečujících nástrojů. TrueCrypt byl vyvinut jako otevřený software, takže je velmi nepravděpodobné, že by se v jeho kódu vyskytovala tzv. zadní vrátka nebo vážné bezpečnostní chyby, aniž by si toho někdo všiml. TrueCrypt umožňuje i šifrování celého systémového oddílu. V tomto ohledu však podporuje jen některé operační systémy (vizte sekci 4.2). Šifrování operačního systému provádí tak, že zašifruje celý systémový disk a uloží na něj svůj vlastní zavaděč, který při startu vyžaduje heslo. Pomocí TrueCryptu je možné vytvářet i tzv. skrytý svazek a na něj lze uložit operační systém.

Tato práce se zaměřuje na obnovu hesel diskových oddílů šifrovaných nástrojem TrueCrypt. Jde o proces, jehož cílem je ze zašifrovaných dat získat heslo. Princip spočívá v tom, že se na vstup dešifrovacího algoritmu vkládají různé kombinace znaků (možná hesla), dokud se data nedešifrují úspěšně. Obnova hesel se využívá i v oblasti forenzní analýzy. Orgány činné v trestním řízení mohou např. obnovovat hesla zašifrovaných dat při vyšetřovaní nějakého trestného činu v případě, že někdo odmítne poskytnout přístup k šifrovaným datům. Nástroj k obnově hesel je možné použít i v případě, kdy někdo zapomene heslo a nutně potřebuje data dešifrovat. Vytvořený modul bude součástí nástroje Fitcrack, který umožňuje obnovu hesel mnoha typů šifrovaných souborů a bude tak rozšiřovat jeho funkcionalitu o další podporu obnovy hesel diskových oddílů šifrovaných TrueCryptem. Výpočet bude možné akcelerovat pomocí GPU a nástroj Fitcrack podporuje i distribuovaný způsob výpočtu.

Kapitola 3 se zabývá generováním náhodných čísel, hešovacími funkcemi a šifrovacími algoritmy, které používá nástroj TrueCrypt. Kapitola 2 se zabývá nástrojem Fitcrack a standardem OpenCL, který Fitcrack využívá k akceleraci výpočtů pomocí GPU. Kapitola 4 přibližuje TrueCrypt, ale zaměřuje se především na TrueCryptem šifrované svazky. A v kapitole 5 je popsán návrh nového modulu Fitcracku, který bude provádět útok na diskové oddíly šifrované TrueCryptem.

Kapitola 2

Fitcrack

Fitcrack je nástroj pro obnovu hesel, který je schopen pracovat samostatně nebo distribuovaně. Tento nástroj je dokáže obnovovat hesla ze šifrovaných dokumentů PDF a MS Office, archivů formátu ZIP, RAR nebo 7-Zip. Obnova hesel takovýchto souborů by na jednom výpočetním stroji byla pomalá a trvala by poměrně dlouho, což by mohlo mít za následek to, že by bylo možné obnovovat jen krátká hesla (např. o délce 3 znaky). Proto je do Fitcracku přidána podpora akcelerace výpočtů pomocí GPU za použití technologií OpenCL a CUDA. Tím se navazuje na původní myšlenku akcelerace v nástroji Wrathion [18, 8]. K distribuovaným výpočtům je použita platforma BOINC, díky které je možné využít výpočetní výkon velkého množství počítačů i v geograficky oddělených lokalitách. Fitcrack byl vyvinut v rámci výzkumné skupiny NES@FIT na Fakultě informačních technologií Vysokého učení technického v Brně.

2.1 Generátory hesel

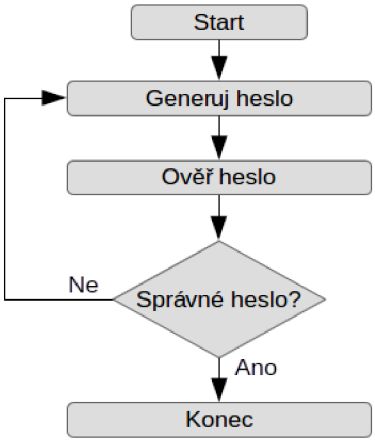
Generování hesel může být velmi výpočetně náročné, protože s délkou hesla se počet kombinací znaků v daném hesle zvyšuje exponenciálně. Proto je důležité, jak velkým výpočetním výkonem disponujeme. Podle toho je možné si spočítat, jaké množství hesel za daný čas je možné zkoušet. Existují různé techniky, jak generovat hesla.

Útok hrubou silou

Útok hrubou silou je ta nejjednodušší metoda jak generovat hesla. Jedná se o útok, kdy se postupně generují všechny možné kombinace znaků hesla. Maximální a minimální počet znaků v hesle může být omezen. Dále musí být ke generování hesel definováno, které znaky se mají použít.

Slovníkový útok

Slovníkový útok je metoda, kdy se používají hesla ze zvoleného slovníku. Slovník představuje seznam hesel, která se mají zkoušet. Z tohoto seznamu lze vybírat jen některá hesla a tím ještě snížit množství zkoušených hesel. Může se jednat o slovník reálně používaných hesel a nebo vygenerovaný slovník podle nějakých pravidel. Slovníkový útok má často daleko menší množství zkoušených hesel než útok hrubou silou, takže se obnovování hesel zpravidla organizuje tím způsobem, že se nejprve provede slovníkový útok a pokud se heslo nenajde, neboť teprve přijde na řadu útok hrubou silou.



Obrázek 2.1: Princip obnovy hesel na CPU

2.2 Obnova hesel

Nejdříve je nutné získat informace o souboru a z nich vybrat ty, které jsou důležité k obnově hesel, například způsob šifrování. K tomu se využívá nástroj FitXtractor. FitXtractor vytvoří soubor XML, ve kterém jsou uloženy metadata nutné k ověření správného hesla. Fitcrack může obnovovat hesla jako samostatná jednotka a nebo může fungovat distribuovaně. Díky tomu se dá využít značné množství výkonu zařízení výpočetní techniky připojených klientů. V případě distribuovaného způsobu výpočtu může správce nastavit maximální počet klientů, kteří se mohou k výpočtu připojit. Metadata získaná FitXtractorem jsou následně rozdělena a připojeným klientům je zasláno, jaká hesla mají zkoušet a další potřebná metadata ke zkoušení obnovy hesel. Výhoda extrahování jen potřebných informací spočívá v tom, že připojená zařízení nemají přístup k citlivým datům a sníží se tím zátěž celé sítě.

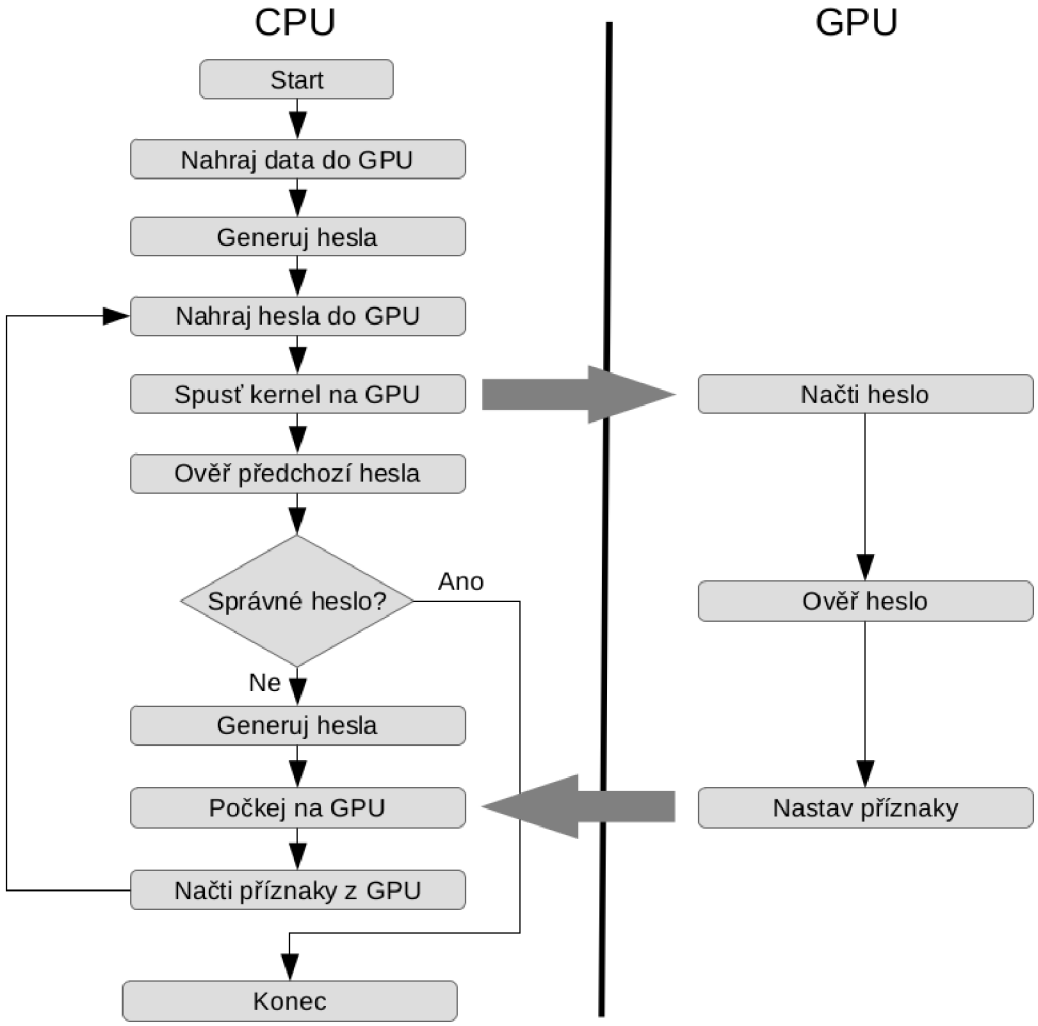
K obnově hesla je nutné stanovit, která se musí vyzkoušet. Hesla ze získají z generátorů, které již byly popsány v sekci 2.1. Zkoušení hesel může být velmi náročný proces, protože se musí vykonat složité výpočty. Naštěstí se tyto operace dají velmi dobře optimalizovat, protože zkoušení různých hesel, na sobě nijak výpočetně nezávisí. Díky tomu je možné tento proces paralelizovat. K paralelizaci (akceleraci) jsou využity GPU, jak už bylo zmíněno dříve.

2.2.1 CPU

Na procesoru se proces obnovy hesel ničím nekomplikuje, protože procesor je vybavený pro vykonávání všech potřebných operací. Princip obnovy hesel na procesoru je znázorněn na obrázku 2.1.

2.2.2 GPU

Obnova hesel na GPU je daleko komplikovanější, protože se celý proces výpočtu musí řídit pomocí procesoru a dále je nutné zajistit přenos potřebných dat na grafickou kartu. Generování hesel není tak výpočetně náročné jako jejich ověřování, protože se při něm nemusí provádět kryptografické algoritmy. Díky tomu je možné si generovat hesla na CPU a až potom spustit zkoušení hesel. Druhou možností je provádění této činnosti pomocí GPU.

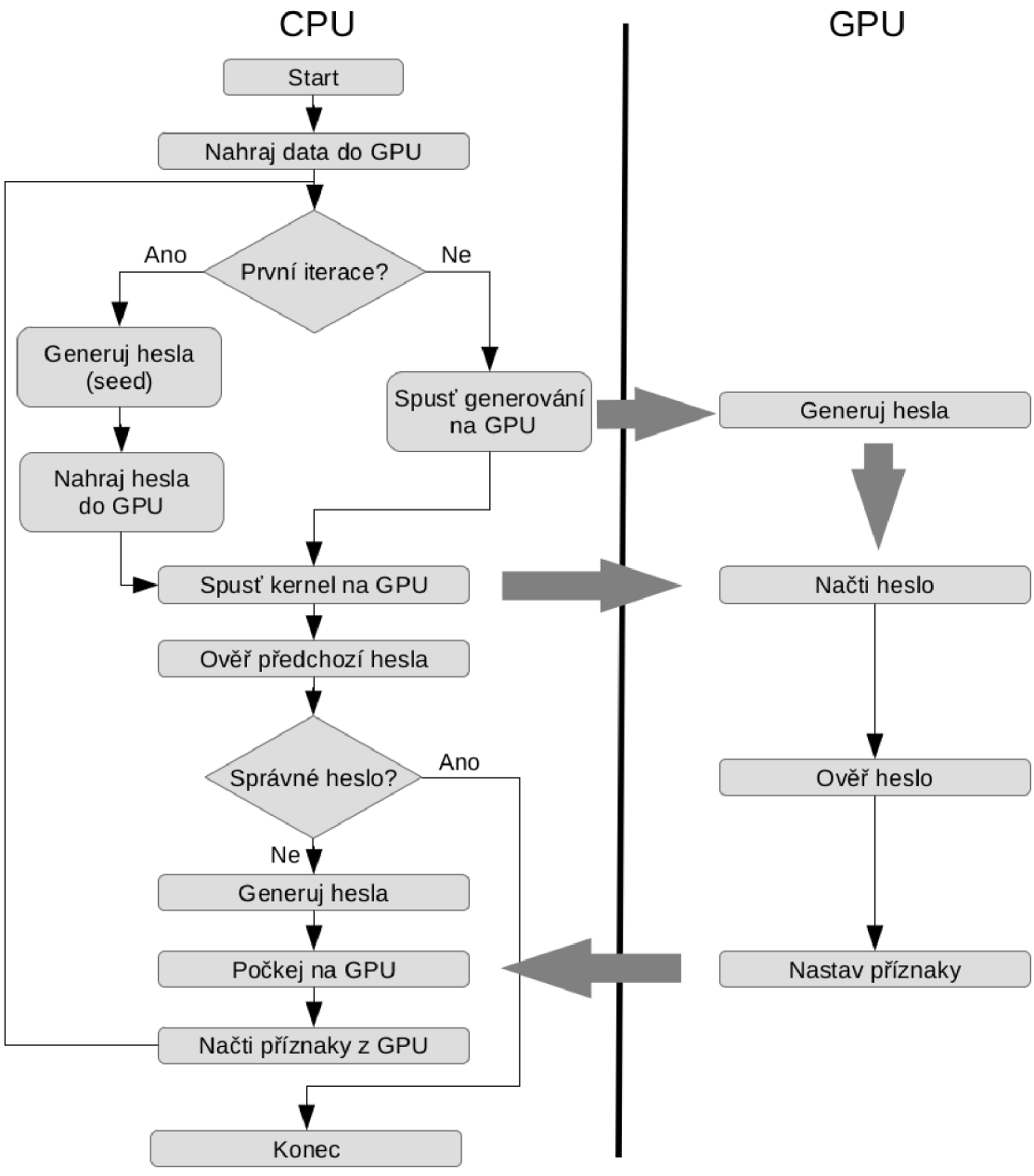


Obrázek 2.2: Princip obnovy hesel s generátorem hesel v CPU

Záleží na tom, zda je generování hesel tak náročné, aby se to muselo provádět na GPU. Princip obnovy hesel s generátorem hesel v CPU je na obrázku 2.2 a princip obnovy hesel s generátorem hesel v GPU se od předchozího liší v tom, že se při každém načtení nových informací o tom, z čeho se mají generovat ještě nevyzkoušená hesla, poté se vygenerují na GPU a dále se všechny používají k obnově hesel. Proces generování na GPU je znázorněn na obrázku 2.3.

2.3 Modularita

Fitcrack je navržen tak, aby bylo možné jednoduše doplňovat nové moduly pro obnovu hesel nových formátů v tzv. modulech. Díky tomu je možné prostřednictvím nových modulů program rozšiřovat. Fitcrack momentálně obsahuje moduly k obnově hesel těchto formátů:



Obrázek 2.3: Princip obnovy hesel s generátorem hesel v GPU

PDF, ZIP, 7z, RAR, DOC, XLS, PPT, DOCX, XLS (novější), PPTX. Podrobný přehled těchto formátů je zveřejněn na oficiálních stránkách tohoto nástroje[1].

2.4 OpenCL

OpenCL (Open Computing Language) [12], je standard pro paralelní programování různých typů procesorů, používaných v osobních počítačích, serverech, mobilních telefonech a vestavěných platformách. Prvotní návrh OpenCL vytvořila společnost Apple. Od roku 2008 je OpenCL spravovaný neziskovým průmyslovým konsorciem Khronos Group [11]. OpenCL popisuje tyto modely chování platforem a zařízení, které mu odpovídají: Model platformy, Výpočetní model, Paměťový model a Programovací model.

2.4.1 Model platformy

Model platformy se skládá z hostitelského systému a k němu je připojeno jedno nebo více zařízení OpenCL. Zařízení OpenCL je rozděleno do jedné nebo více výpočetních jednotek, které se skládají z jednoho nebo více výpočetních prvků. Výpočty na tomto zařízení se provádí pomocí těchto výpočetních prvků [12].

2.4.2 Výpočetní model

Výpočetní model se skládá ze dvou částí: Výpočetní část (tzv. kernel), která se vykonává na jednom nebo více výpočetních jednotkách OpenCL, a hostitelský program, ten se stará o komunikaci s jednotlivými výpočetními jednotkami.

Výpočetní jednotky zpracovávají tu část programu, která byla psána v OpenCL C (dále jen program). Program se skládá z jednoho nebo více výpočetních vláken. Tato vlákna jsou instancemi funkčního objektu, který se nazývá kernel. Při spuštění výpočtu specifikuje aplikace jedno až třírozměrný indexový prostor, celkový počet vláken a případně velikost skupin do kterých se budou tyto instance sdružovat. Z těchto informací OpenCL následně určí počet skupin a každému vláknu přiřadí globální index, lokální index a skupinový index. Lokální index identifikuje instanci kernelu v rámci skupiny. Globální index identifikuje instanci kernelu v rámci všech instancí. Skupinový index je identifikátor skupiny do níž instance kernelu patří [12].

2.4.3 Paměťový model

Paměťový model popisuje strukturu, obsah a chování paměti, kterou využívá platforma OpenCL při běhu programu. Tento model specifikuje jak jsou data uložená v paměti a jak se s nimi pracuje, když se vykonává program. Paměť se dělí na tyto čtyři typy: globální paměť, paměť konstant, lokální paměť a soukromá paměť.

Do globální paměti mohou číst i zapisovat instance kernelu. Hostitelský program může také číst i zapisovat do této paměti, ale do tzv. rour, které jsou také součástí globální paměti, přístup nemá. Z paměti konstant může číst hostitelský program i instance kernelu, ale jen hostitelský program do ní může i psát. Do lokální paměti mají přístup (čtení i zápis) jen instance kernelu, kterým byla přidělena a je viditelná všem instancím kernelu ve stejné skupině. Do soukromé má přístup jen instance, které byla přidělena[12].

[1] http://wrathion.fit.vutbr.cz/fitcrack/?i=features

2.4.4 Programovací model

OpenCL podporuje úlohově paralelní nebo datově paralelní programovací modely. Datově paralelní programovací model definuje společný běh instancí stejného kernelu, které zpracovávají datové složky vstupní datové struktury. V nejjednodušším případě případně jedna instance kernelu na jednu datovou složku. Oproti tomu úlohově paralelní programovací model umožňuje spouštět několik instancí různých kernelů. Neumožňuje ale spuštění v jednu chvíli několika instancí jednoho kernelu tak, aby spolu mohly komunikovat [12].

Kapitola 3

Kryptografické algoritmy

V této kapitole se budeme zabývat generováním náhodných čísel, hešovacími algoritmy, šifrovacími algoritmy a režimy blokových šifer, které používá TrueCrypt. Šifrování je transformace vstupního řetězce (tzv. otevřeného textu) na výstupní řetězec (tzv. šifrovaný text).

3.1 Generátor náhodných čísel

Generátor náhodných čísel se používá ke generování soli, šifrovacího klíče, sekundárního klíče, kvůli režimu XTS (vizte dále), a tzv. Keyfile. Kryptografická sůl je několik náhodných bitů (bytů), které slouží jako doplňující vstup při použití jednosměrné funkce. Vznikla za účelem ochrany proti tzv. Duhovému útoku, který používá ke zpětnému vyhledávání zpráv z předpočítaných tabulek s otisky. Při generování jsou mimo jiné využita data z následujících zdrojů:

- pohyb myši,
- stisknutí kláves,
- hodnoty generované z vestavěného generátoru náhodných čísel (/dev/random a /dev/urandom, jen Mac OS X a Linux),
- MS Windows CryptoAPI (ze kterého se získávají v intervalu 500 ms, jen MS Windows),
- statistiky síťového rozhraní (NETAPI32, jen MS Windows),
- různé rukojeti Win32, systémové proměnné času a čítače (ze kterých se čísla získávají v intervalu 500 ms, jen MS Windows).

Podrobněji je generování náhodných čísel popsáno v dokumentaci TrueCryptu [20].

3.2 PBKDF2

PBKDF2 je funkce pro odvození klíče. K tomu, aby bylo možné ji popsat je nutné nejdřív definovat některé pojmy. Kryptografická sůl je několik náhodných bitů (bytů), které slouží jako doplňující vstup při použití jednosměrné (v tomto případě hešovací) funkce kde umožňuje, aby její výstup měl mnoho možných variant. Funkce pro odvození klíče PBKDF2 má pět funkčních parametrů:

$DK = PBKDF2(PRF, Password, Salt, c, dkLen)$

PRF je hešovací funkce, která má typicky dva parametry: uživatelské heslo a kryptografickou sůl. Z toho vyplývá, že i funkce PBKDF2 musí mít kryptografickou sůl (Salt) a uživatelské heslo (Password) ve svých parametrech. Parametr c je počet iterací, které se mají provést. Parametr dkLen je požadovaná délka odvozeného klíče. Funkce vrací DK, což je odvozený klíč délky dkLen. V průběhu vykonávání funkce PBKDF2 se každou iterací vypočítá blok T_i, ve kterém je odvozený klíč. Ten je vypočítán takto:

$DK = T_1||T_2||...||T_{dklen/hlen}$

$T_i = F(Password, Salt, c, i)$

Při každé iteraci i se tedy volá funkce F. Funkce F při každé iteraci zavolá funkci PRF a provede xor výsledku se xorem předchozích výsledků těchto volání funkce PRF, pokud se nejedná o první volání. První volání funkce PRF dostává parametry Password a sůl, která je konkatenovaná s i. To i je kódované jako 32-bitové celé číslo ve formátu Big endian. V následujících voláních dostává funkce PRF parametry Password a místo soli xor předchozích volání PRF.

$F(Password, Salt, c, i) = U_1 \hat{} U_2 \hat{} ... \hat{} U_c$

kde:

$U_1 = PRF(Password, Salt||INT\_32\_BE(i))$

$U_2 = PRF(Password, U_1)$

...

$U_c = PRF(Password, U_{c-1})$

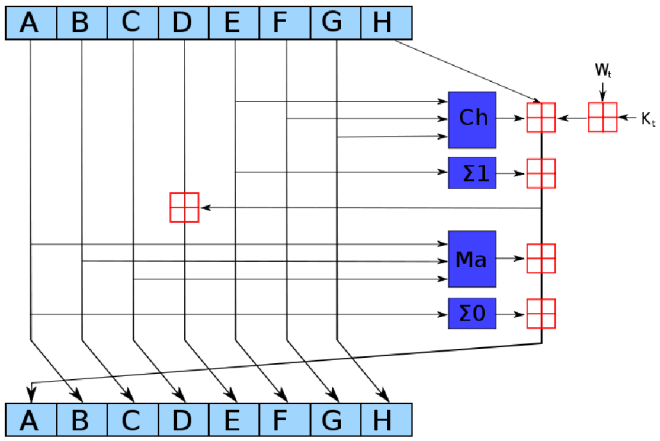
Obtížnost útoku hrubou silou se zvyšuje s počtem iterací. Praktickým limitem počtu iterací je neochota uživatelů tolerovat znatelné zpoždění při přihlášení k počítači nebo doba zobrazení zprávy o dešifrování zprávy. Použití soli brání útočníkům v použití před počítaných slovníků odvozených klíčů [10].

3.3 Hešování

Kryptografické hešovací funkce mají velmi důležitou roli v oblasti moderní kryptografie. Hešovací funkce obecně převádí vstup (tzv. zprávu), která může být různé délky na výstup (tzv. hešový kód nebo otisk) fixní délky. Pro hešovací funkci h, zprávu D a otisk R tedy platí: $h : D \rightarrow R$ a $|D| > |R|$. Ideální kryptografická hešovací funkce je jednosměrná. Jednosměrnost znamená, že pro daný otisk c je nemožné spočítat x takové, aby platilo h(x) = c. V praxi se to projevuje tak, že je snaha vytvořit takovou funkci, u které je tento výpočet co nejobtížnější. Další vlastnost ideální kryptografické hešovací funkce je bezkoliznost. Protože je délka vstupu hešovací funkce větší než délka výstupu, je nemožné, aby neexistovaly žádné kolize. Proto u reálné hešovací funkce je důležité, aby nalezení dvojice vstupů (x,y), pro které platí h(x) = h(y) bylo co nejobtížnější. Dalším pravidlem je, že musí být co nejtěžší najít pro vstup x takové y, aby platilo h(x) = h(y). TrueCrypt používá tyto hešovací algoritmy k ochraně hlavičky:

- RIPEMD-160,
- SHA-512,
- Whirlpool.

TrueCrypt specifikoval počty iterací pro odvození klíče takto: 2000 iterací pro Ripemd-160, 1000 iterací pro SHA-512 a 1000 iterací pro Whirlpool. Pro funkci PBKDF2 je nutné, aby



Obrázek 3.1: Kompresní funkce SHA-512

se všechny hešovací funkce vypočítaly pomocí HMAC[1]. Funkce HMAC je definována jako: $HMAC(K, m) = H((K' \oplus opad) \parallel ((K' \oplus ipad) \parallel m))$, kde H znamená kryptografickou hešovací funkci, K je klíč, m je zpráva, K' odvozený klíč z klíče K, $\parallel$ znamená konkatenaci, $\oplus$ znamená xor, opad je vnější výplň (0x5c5c5c...5c5c, hexadecimální konstanta velikosti jednoho bloku), ipad je vnitřní výplň (0x363636...3636, hexadecimální konstanta velikosti jednoho bloku) [13].

3.3.1 SHA-512

SHA-512 je hešovací algoritmus, který byl navržen NSA[2] a zveřejněn NIST[3](v FIPS PUB 180-2[4]) v roce 2002 (první návrh byl zveřejněn v roce 2001). Výstup tohoto algoritmu má velikost 512 bitů. Na obrázku 3.1[5] je znázorněna jedna iterace kompresní funkce SHA-512. Modré komponenty provádí tyto operace:

$\mathbf{Ch}(E, F, G) = (E \wedge F) \oplus (\neg E \wedge G)$

$\mathbf{Ma}(A, B, C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$

$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$

$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$

$\boxplus$ představuje součet modulo[6] 2^{64}, $\ggg$ představuje rotaci doprava a $\oplus$ znamená xor.

3.3.2 RIPEMD-160

RIPEMD-160 je hešovací funkce založená na kryptografickém hešovacím algoritmu MD4, při jejím vývoji byly brány v úvahu znalosti získané z analýzy kryptografických hešovacích algoritmů MD4, MD5 a RIPEMD [15].

Hešovací funkce RIPEMD-160 byla navržena Hansem Dobbertinem, Antoonem Bosselaersem a Bartem Preneelem v RIPEMD-160: A Strengthened Version of RIPEMD, aby

[1] Keyed-Hashing for Message Authentication

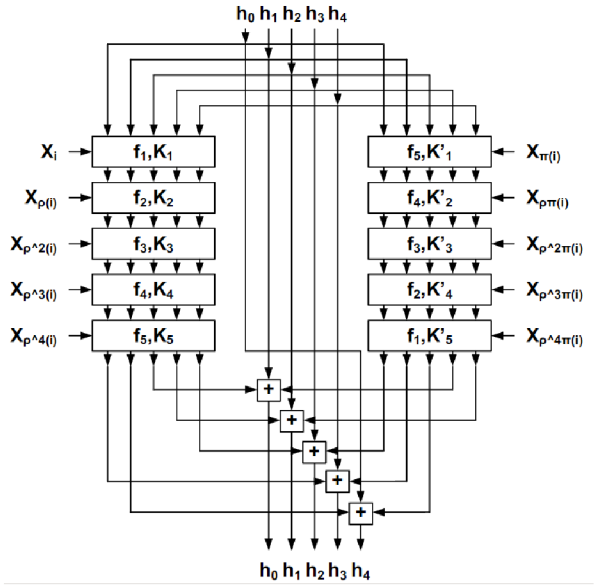
[2] Národní bezpečnostní agentura je vládní kryptologická organizace USA.

[3] Národní institut standardů a technologie je laboratoř měřicích standardů při ministerstvu obchodu USA.

[4] https://www.coursehero.com/file/14958402/FIPS-180-2-SHApdf

[5] Obrázky převzaty z https://en.wikipedia.org/wiki/SHA-2

[6] Obrázky převzaty z https://en.wikipedia.org/wiki/SHA-2



Obrázek 3.2: Kompresní funkce RIPEMD-160

nahradila RIPEMD. Jedná se o iterativní hešovací funkci, která převádí 512-bitovou zprávu ze vstupu na 160-bitový otisk. Stejně jako její předchůdce RIPEMD se skládá ze dvou paralelních toků. Samotný proces výpočtu hešovací funkce se skládá z následujících dvou částí: rozšíření zprávy a transformace aktualizace stavu [15].

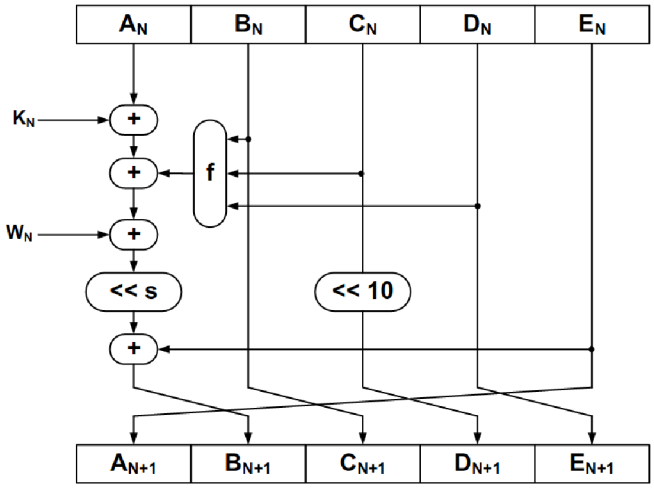
Rozšíření zprávy probíhá tak, že v každém kole se slova zprávy obmění. Pro levý a pravý tok jsou použity různé obměny. Obměny jsou znázorněny na obrázku 3.2 [15].

Aktualizace stavu začíná od pevně dané počáteční hodnoty (inicializačního vektoru) pěti 32-bitových registrů. Aktualizuje je v pěti kolech, které se skládají z 16 kroků. Každý je transformován podle rozšíření zprávy w_i v kroku i. Na obrázku 3.3 je znázorněn jeden krok aktualizace stavu. Funkce f je v každém kole jiná. V levém toku je v kole j je použita funkce f_j a v pravém toku je použita funkce f_{6-j} $(j = 1, ..., 5)$. V každém kroku se přičte konstanta K_j, která také závisí na j. Také rotační hodnoty s jsou různé pro každý krok a tok. Po posledním kroku transformace stavu se počáteční hodnoty a hodnoty levého a pravého toku modulárně sečtou, z čehož vznikne finální hodnota. Tato hodnota může být finálním otiskem a nebo počáteční hodnotou (aktualizovaným inicializačním vektorem) v dalším bloku zprávy [15]. Funkce f_j jsou definovány takto[7]:

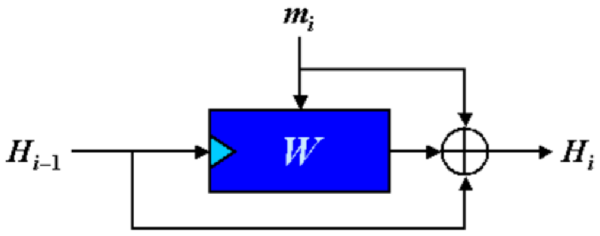
$f(j, x, y, z) = x \oplus y \oplus z$, kde $(0 \leq j \leq 15)$
$f(j, x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$, kde $(16 \leq j \leq 31)$
$f(j, x, y, z) = (x \vee \neg y) \oplus z$, kde $(32 \leq j \leq 47)$
$f(j, x, y, z) = (x \wedge z) \vee (y \wedge \neg z)$, kde $(48 \leq j \leq 63)$
$f(j, x, y, z) = x \oplus (y \vee \neg z)$, kde $(64 \leq j \leq 79)$

3.3.3 Whirlpool

Algoritmus Whirlpool byl navržen Vincentem Rijmenem (spoluautorem šifrovacího algoritmu AES) a Paulem S. L. M. Barretem. Velikost výstupu tohoto algoritmu je 512 bitů. První verze Whirlpoolu, kterému se dnes říká Whirlpool-0, byl zveřejněn v listopadu roku



Obrázek 3.3: Krok RIPEMD-160



Obrázek 3.4: Whirlpool

2000. Druhá verze, která se jmenuje Whirlpool-T byla v projektu NESSIE[7] byla vybrána do portfolia silných kryptografických primitiv. TrueCrypt používá třetí (finální) verzi algoritmu Whirlpool, který byl zaveden ISO[8] v mezinárodním standardu ISO/IEC 10118-3:2004.

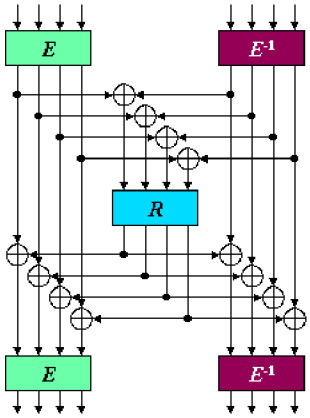
Algoritmus Whirlpool je Mrkleovo-Damårdovou konstrukcí[9] nad blokovou šifrou W, což je upravený šifrovací algoritmus AES v módu Miyaguchi–Preneel. Tato základní struktura je vyobrazena na obrázku 3.4 [17, 2].

Bloková šifra W se skládá ze stavové matice bytů S, která má rozměry 8×8, dává dohromady tedy celkem 512 bitů. Proces šifrování se provádí v deseti kolech. V každém kole se provedou těmito čtyřmi funkcemi: SubBytes (SB), ShiftColumns (SC), MixRows (MR) a AddRoundKey (AK). Vzorec výpočtu: $S = AK \circ MR \circ SC \circ SB(S)$ Záměna bytů (SubBytes) nahradí každý stavový byte jiným bytem nezávisle na ostatních bytech podle tzv. S-boxu. S-boxu je typicky vyhledávací tabulka, ve které se vyhledává, čím se má daný byte nahradit. V tomto případě je 8-bitový S-box zakomponován do 4-bitových S-boxů, které se jmenují $Emini - box$, $E^{-1}mini - box$ a $Rmini - box$. S-box je implementován rekurzivně, vizte obrázek 3.5 [17]. ShiftColumns cyklicky posouvá každý bajt v každém stavovém sloupci.

[7] NESSIE (New European Schemes for Signatures, Integrity and Encryption), byl projekt organizovaný Evropskou Unií, podobný schvalovací proceduře AES.

[8] Mezinárodní organizace pro normalizaci.

[9] Merkleova-Damgårdova konstrukce je pojem z oboru kryptografie, kterým se označuje konkrétní způsob vytváření kryptografických hešovacích funkcí z vhodných jednosměrných kompresních funkcí. Více informací na: http://cseweb.ucsd.edu/~mihir/papers/gb.pdf



Obrázek 3.5: S-box Whirlpoolu

Sloupec j je posunutý dolů o j pozicí. Funkce MixRows provádí pravé násobení každého řádku s maticí o rozměrech 8×8. AddRoundKey pomocí bitového xoru připočítá vypočítaný klíč nazvaný round key, který je vypočítaný pomocí stejných operací jako stavová matice, avšak operace AddRoundKey je nahrazena operací AddRoundConstant, což pouze přičítá předurčenou konstantu. Vizte obrázek 3.6 [17].

3.3.4 CRC32

Cyklický redudantní součet CRC (Cyclic redundancy check) [4] je technika používaná k detekci chyb v datech. Tato technika však ale nestačí k tomu, aby bylo možné detekované chyby opravit. Metoda k datům přidává určitý počet kontrolních bitů, kterým se říká kontrolní součet. Při ověřování správnosti dat se vypočítá kontrolní součet z datové části a porovná s kontrolním součtem, který byl k datové části přidán. Pomocí výpočtu takovým způsobem lze ověřit, že s vysokou pravděpodobností nedošlo chybě. Díky tomu, že je tato metoda jednoduše implementovatelná a používá rychle a jednoduše proveditelné operace pomocí současného hardwaru (2017), je po světě velmi rozšířená.

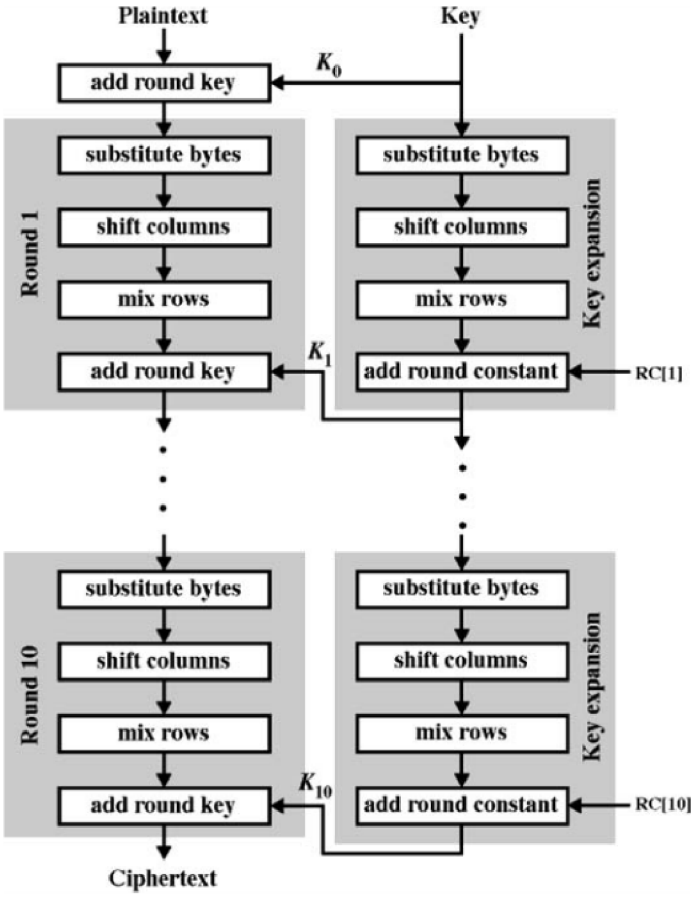
3.4 Šifrovací algoritmy

Všechny šifrovací algoritmy využívané TrueCryptem pracují v režimu XTS popsaném dále a jejich přehled je uveden v tabulce 3.1. Tabulka je platná pro verzi 7.1a [20].

3.4.1 AES

Advanced Encryption Standard (AES) specifikuje šifrovací algoritmus Rijndael, který byl navržen Joanem Daemenem a Vincentem Rijmenem. Byl zveřejněn v roce 1998. FIPS[10] ho schválila k tomu, aby byl používán k ochraně citlivých dat federálních ministerstev a agentur Spojených Států Amerických. TrueCrypt používá AES se 14 koly a 256-bitovým klíčem (tj. AES-256, zveřejněný v roce 2001) pracující v režimu XTS, vizte sekci Režimy blokových šifer [20].

[10]FIPS je organizace pro standardy USA vyvinuté vládou USA.



Obrázek 3.6: Bloková šifra W

Algoritmus	**Návrháři**	**Velikost klíče (Bity)**	**Velikost datového bloku (Bity)**	**Režim**
AES	J. Daemen, V. Rijmen	256	128	XTS
Serpent	R. Anderson, E.Biham, L. Knudsen	256	128	XTS
Twofish	B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson	256	128	XTS
AES-Twofish		256; 256	128	XTS
AES-Twofish -Serpent		256; 256; 256	128	XTS
Serpent-AES		256; 256	128	XTS
Serpent-Twofish -AES		256; 256; 256	128	XTS
Twofish-Serpent		256; 256	128	XTS

Tabulka 3.1: Šifrovací algoritmy a jejich konkatenace

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

Obrázek 3.7: Matice pro MixColumns

Struktura algoritmu

Provedení operací počátečního kroku:

- AddRoundKey.

Provedení kol, které se skládají z následujících operací:

- SubBytes,
- ShiftRows,
- MixColumns,
- AddRoundKey,

Provedení posledního kola, které se skládá z těchto operací:

- SubBytes,
- ShiftRows,
- AddRoundKey.

Záměna bytů (SubBytes) probíhá tak, že každý bajt bloku je nahrazen jiným bajtem podle S-boxu. S-box je vyhledávací tabulka, ve které se vyhledává, čím se má daný byte nahradit. Bajty jsou uspořádány do matice. Posun řádků (ShiftRows) provede posuny těchto řádků. Kombinování sloupců (MixColumns) provede násobení sloupce s předem danou maticí (3.7[11]). Přidání podklíče (AddRoundKey) provede xor s podklíčem, což je stejně velká matice jako matice šifrovaných bytů. Podklíč je pro každé kolo vypočítán pomocí tzv. AES key schedule, který je podrobně popsán například ve FIPS 197 [16].

3.4.2 Serpent

Algoritmus Serpent byl navržen Rossem Andersonem, Eli Bihamem a Larsem Knudsenem v roce 1998. Používá 256-bitový klíč, 128-bitové bloky a pracuje v režimu XTS [20]. Dále byl kandidátem při výběru algoritmu do standardu AES. Největší rozdíl mezi algoritmy AES (tehdy Rijndael) a Serpent je v počtu iterací. Serpent jich má více, čímž se stává pomalejším, ale bezpečnějším algoritmem.

Algoritmus je navržen jako substitučně-permutační síť, která zpracovává 128-bitový blok ve 32 rundách (nebo reiteracích stejného algoritmu). Každá runda zpracovává tedy 4 bity. Každá runda použije jeden z osmi 4 bitových S-boxů[12], které se používají 32 krát paralelně. Byl vyvinut tak, že všechny operace mohou být provedeny paralelně pomocí 32 1 bitových úseků, tím se maximalizuje paralelismus [14].

[11] Obrázek převzat z https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[12] S-box je typicky vyhledávací tabulka, ve které se vyhledává, čím se má daný byte nahradit.

3.4.3 Twofish

Algoritmus Twofish byl navržen Brucem Schneierem, Johnem Kelsey, Dougem Whitingem, Davidem Wagnerem, Chrisem Hallem a Nielsem Fergusonem. Zveřejněn byl v roce 1998. Používá 256-bitový klíč, 128-bitové bloky a pracuje v režimu XTS. Twofish byl jedním z kandidátů při výběru algoritmu do standardu AES. Šifra používá S-boxy[13], které jsou závislé na klíči. Twofish se dá chápat jako kolekce 2128 kryptosystémů, kde 128 bitů, odvozených z 256-bitového klíče řídí výběr kryptosystému [20, 19].

3.4.4 Konkatenace algoritmů

TrueCrypt používá těchto pět způsobů konkatenace algoritmů AES, Serpent a Twofish:

- AES-Twofish,
- Serpent-AES,
- Twofish-Serpent,
- AES-Twofish-Serpent,
- Serpent-Twofish-AES.

Například AES-Twofish znamená, že datový blok bude šifrován nejprve algoritmem Twofish a následně algoritmem AES.

3.5 Režimy blokových šifer

Nejdřív TrueCrypt používal režim CBC. Ve verzích 4.1 až 4.3a používal režim LRW. Dnes používá režim XTS, který je považován za bezpečnější, dříve používané režimy CBC a LRW [20].

3.5.1 XTS

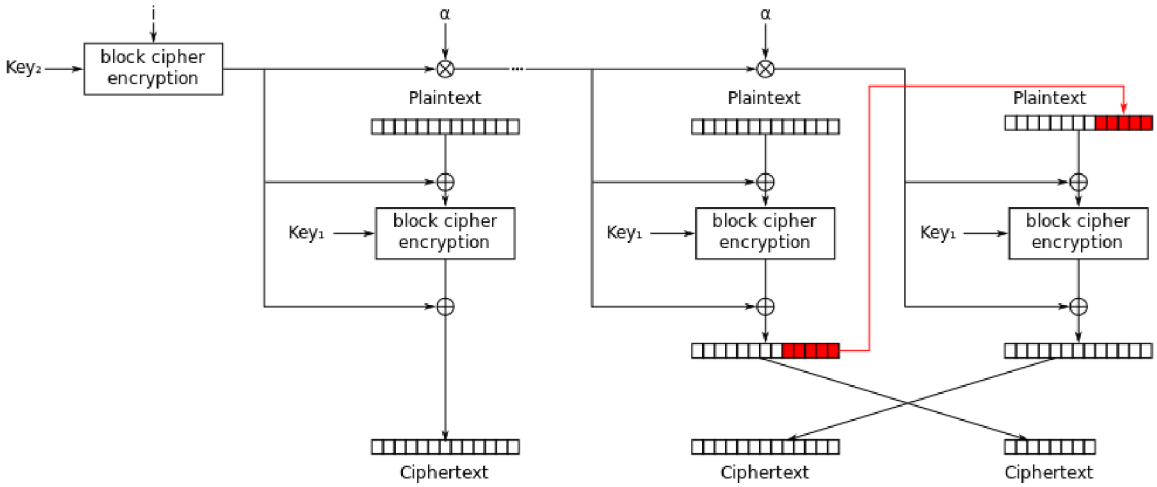
Režim XTS byl navržen Phillipem Rogaway v roce 2003. Ve skutečnosti jde o paralelu režimu XEX, jen s drobnými změnami. Režim XEX používá jediný klíč pro dva různé účely, zatímco režim XTS používá dva nezávislé klíče. V roce 2010 byl režim XTS schválen NIST[14] pro ochranu důvěrnosti dat na paměťových zařízeních. V roce 2007 byl IEEE[15] schválen pro kryptografickou ochranu dat na blokově orientovaných paměťových zařízeních (IEEE 1619) [20]. Režim XTS se je popsán touto rovnicí:

$C_i = EK_1(\pi\hat{}(EK_2(n) \otimes \alpha i))\hat{}(EK_2(n) \otimes \alpha i)$

Kde:

- $\otimes$ znamená násobení dvou polynomů nad binární pole GF (2) modulo x128 + x7 + x2 + x + 1.
- K1 je šifrovací klíč (256-bit pro každou podporovanou šifru, tj. AES, Serpent a Twofish).

[13] S-box je typicky vyhledávací tabulka, ve které se vyhledává, čím se má daný byte nahradit.
[14] Národní institut standardů a technologie je laboratoř měřicích standardů při ministerstvu obchodu USA.
[15] Institut pro elektrotechnické a elektronické inženýrství



Obrázek 3.8: Princip režimu XTS

- K2 je sekundární klíč (256-bit pro každou podporovanou šifru, tj. AES, Serpent a Twofish).
- Číslo i je index šifrovaného bloku v rámci datové jednotky; začíná od nuly.
- Číslo n je index datové jednotky v rozsahu K1; začíná od nuly.
- α je primitivní element Galoisova tělesa (2128) [3], které odpovídá polynomu x (tj., 2).

Velikost každé datové jednotky je vždy 512 bytů (bez ohledu na velikost sektoru). Princip režimu XTS je znázorněn na obrázku 3.8[16].

[16]Obrázek převzat z https://en.wikipedia.org/wiki/Disk_encryption_theory

Kapitola 4

TrueCrypt

TrueCrypt je nástroj pro vytváření a údržbu šifrovaných diskových oddílů a ty jsou šifrovány za běhu. To znamená, že se data šifrují těsně před tím, než se uloží na disk. Hned potom, co se nějaká data z disku přečtou, dešifrují se, a to bez jakéhokoliv zásahu uživatele. TrueCrypt umožňuje uživateli vytvoření šifrovaného svazku, který je uložený v souboru, šifrování již existujícího svazku a nebo šifrování svazku operačního systému [20, 6].

4.1 Šifrovaný svazek

Šifrovaný svazek nemá žádnou signaturu[1]. Celý se jeví jako změť náhodných čísel. Šifrovaný svazek se skládá z hlaviček a datové oblasti. Datová oblast je celá zašifrovaná a klíč k odšifrování je uložen v hlavičce. Hlavička je rovněž šifrovaná, ale svým vlastním klíčem. Jediná nešifrovaná část hlavičky je sůl[2], ta je v tomto případě 64 bytů náhodně vygenerovaných čísel, která se generovala při vytváření svazku. Bez hesla tedy nelze zjistit ani jakým způsobem byl svazek šifrován. Hlavičky mohou být až čtyři. Jedna pro normální svazek, jedna pro tzv. skrytý svazek (vizte dále) a dvě jako jejich zálohy [20].

K šifrování jsou použity dva náhodně generované klíče, náhodně generovaná sůl, uživatelské heslo a tzv. Keyfile. Šifrovací klíč k datové oblasti je uložen v hlavičce. Připojení svazku tedy znamená získání šifrovacího klíče k datové oblasti z hlavičky. Vzhledem k tomu, že je hlavička šifrovaná pomocí klíče hlavičky a uživatelského hesla (případně i Keyfile), tak k přístupu k datům je nutné pouze úspěšně dešifrovat hlavičku.

4.1.1 Skrytý svazek

Uvnitř normálního šifrovaného svazku se může nacházet i skrytý svazek. Jeho hlavička je uložena s posuvem 65536 bytů od začátku hostitelského svazku. Myšlenka skrytého svazku spočívá v tom, že nikdo nemůže vědět zda existuje. Pokud by někdo naléhal na uživatele TrueCryptu, aby mu prozradil heslo, tak tento uživatel může říct heslo k normálnímu svazku a o skrytém svazku pomlčet a ten, kdo na něj naléhá, si nemůže nijak prověřit, zda neexistuje ještě skrytý svazek [20].

Protože skrytý svazek je vždycky uvnitř normálního svazku a nikde není zmínka o jeho existenci, je nutné ho chránit před souborovým systémem v normálním svazku. Pokud by totiž nebyl chráněn, mohlo by nastat, že se při používání normálního (hostitelského) svazku přepíšou data skrytého svazku. Pokud chce uživatel ochránit skrytý svazek, musí připojit

[1] Signatura je označení souboru, např. řetězec ELF na 2-4. bytu ve spustitelném souboru formátu ELF.
[2] Vizte sekci 3.1.

normální svazek v hostitelském módu a zadat i heslo skrytého svazku. TrueCrypt následně připojí normální svazek a dešifruje hlavičku skrytého svazku, kde jsou uložené informace o velikosti skrytého svazku. Díky tomu je možné zajistit, aby se žádná data skrytého svazku nepřepsala.

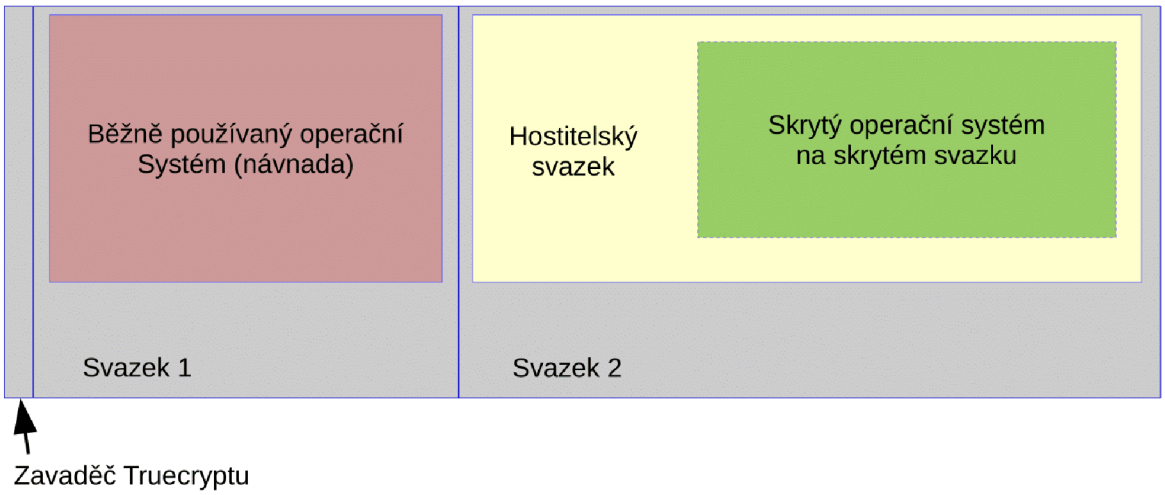
4.2 Šifrovaný operační systém

TrueCrypt umožňuje šifrování celého systémového svazku. To je provedeno tak, že se zašifruje celý systémový svazek a na disk uloží svůj vlastní zavaděč. Po zapnutí počítače naběhne zavaděč TrueCryptu, který vyžaduje heslo šifrovaného svazku. Jakmile dostane heslo, tak dešifruje systémový svazek a spustí operační systém. TrueCrypt podporuje šifrování těchto operačních systémů [20]:

- Windows 7 (32-bit and 64-bit),
- Windows Vista,
- Windows Vista x64 (64-bit) Edition,
- Windows XP,
- Windows XP x64 (64-bit) Edition,
- Windows Server 2008 R2 (64-bit),
- Windows Server 2008,
- Windows Server 2008 x64 (64-bit),
- Windows Server 2003,
- Windows Server 2003 x64 (64-bit),
- Windows 2000 SP4,
- Mac OS X 10.7 Lion (64-bit and 32-bit),
- Mac OS X 10.6 Snow Leopard (32-bit),
- Mac OS X 10.5 Leopard,
- Mac OS X 10.4 Tiger
- Linux (32-bit and 64-bit versions, kernel 2.6 or compatible).

4.2.1 Skrytý operační systém

TrueCrypt umožňuje vytvoření skrytého svazku, na kterém se nachází skrytý operační systém. Na disku se ale musí nacházet nešifrovaný zavaděč. Z toho se dá usoudit, že se na disku nachází šifrovaný operační systém. Je tedy nutné mít operační systém na normálním šifrovaném svazku, aby uživatel měl vysvětlení, proč se na disku nachází zavaděč TrueCryptu. Uživatel by měl při běžném používání počítače používat operační systém na normálním svazku, aby nebylo podezřelé, že počítač používá strašně málo. Skrytý operační systém by měl používat jen pokud by potřeboval pracovat se skrytými daty. Aby se předešlo poškození



Obrázek 4.1: Skrytý operační systém

skrytého operačního systému, musí být skrytý svazek operačního systému být uvnitř jiného svazku, než je svazek, na kterým je běžně používaný operační systém. Hesla tedy budou nakonec tři. Struktura výše zmíněných svazků je vykreslena na obrázku 4.1 [20].

4.3 Hlavička

V hlavičce jsou uloženy všechny informace o šifrovaném svazku včetně klíče k odšifrování datové oblasti tzv. Master Key. Hlavička je sama o sobě šifrovaná. Jediná nešifrovaná část je prvních 64 bytů, kde je uložena kryptografická sůl. Dnes má používaná část hlavičky 512 bytů. Struktura hlavičky je popsána v tabulce 4.1. Dříve se ale používalo všech 65536 bytů. Těch dnes nepoužívaných 65024 bytů bylo rezervovaných pro případ šifrování operačního systému. Hlavička skrytého svazku má tedy ofset 65536. Zálohy hlaviček jsou velké 65536 bytů i když je používaných jen 512 bytů. Zálohy hlaviček jsou šifrované pomocí své vlastní soli. Porovnáváním hlavičky s její zálohou tedy není možné zjistit, zda šifrovaný svazek existuje nebo zda existuje i skrytý svazek. Umístění skryté hlavičky a záložních hlaviček je v tabulce 4.2. S je velikost hostitelského svazku [20].

4.4 Datová oblast

Celá datová oblast je šifrovaná a klíč k odšifrování je uložen v hlavičce. Tento klíč může mít různou délku. Záleží na tom, které šifrovací algoritmy jsou použity. Kromě toho se vždy pracuje v režimu XTS (vizte sekci 3.5.1), proto jsou klíče minimálně dva. Tyto klíče jsou uloženy ve formě jejich konkatenace v hlavičce s offsetem 256 bytů, kde je pro ně místo 256 bytů. Maximální délka konkatenovaných klíčů je ale jen 192 bytů. Tyto klíče jsou náhodně generovány při vytváření šifrovaného svazku. Po dešifrování hlavičky je možné z ní vyčíst informace o velikosti šifrovaného svazku, jakými algoritmy byl šifrován, velikost sektoru svazku a hlavně klíč k odšifrování svazku [20].

Offset	Velikost (Byty)	Popis
0	64	Sůl
64	4	ASCII řetězec "TRUE"
68	2	Verze hlavičky (5)
70	2	Minimální verze potřebná k otevření svazku
72	4	CRC dešifrovaných bajtů 256-511
76	16	Rezervované
92	8	Velikost skrytého svazku, 0 pro neskrytý
100	8	Velikost svazku
108	8	Bytový offset začátku rozsahu Master Key
116	8	Velikost šifrované rozlohy v rozsahu Master Key
124	4	Bity značící typ šifrování
128	4	Velikost sektoru v bytech
132	120	Rezervované
252	4	CRC dešifrovaných bajtů 64-251
256	256	Konkatenovaný Master Key
512	65024	Rezervované (k šifrovanému operačnímu systému, již nepoužívané)

Tabulka 4.1: Struktura hlavičky

Offset	Velikost (Byty)	Popis
65536	65536	Prostor pro hlavičku skrytého svazku (pokud neexistuje skrytý svazek v tomto svazku, tak obsahuje náhodná čísla). Vizte byty 0-65535.
131072	256	Konkatenovaný Master Key. Offset může být jiný, záleží na systémovém svazku
S-131072	65536	Záloha hlavičky (šifrovaná s jiným klíčem hlavičky, který je vypočítán jinou solí)
S-65535	65536	Záloha hlavičky skrytého svazku (opět s jinou solí, pokud neexistuje, tak jsou tam náhodná čísla).

Tabulka 4.2: Ostatní hlavičky

4.5 Keyfile

Ke zvýšení bezpečnosti šifrovaného svazku je možné při vytváření svazku použít tzv. Keyfile. Keyfile může být jakýkoliv soubor nebo je možné ho vytvořit pomocí generátoru náhodných čísel TrueCryptu. Pokud uživatel zvolí soubor, který je větší než 1 MB, použije se z něj jen 1 MB dat. Při použití Keyfile se zvyšuje bezpečnost tak, že se pomocí předem daného algoritmu z Keyfile a uživatelského hesla vypočítá nové heslo, které je potom použito při volání PBKDF2 [20].

4.6 Ostatní nástroje k šifrování diskových oddílů

VeraCrypt šifrovací nástroj, který je postavený na TrueCryptu. VeraCrypt řeší některé zranitelnosti a bezpečnostní problémy nalezené v TrueCryptu. Například nízký počet iterací při používání PBKDF2. Dále používá stejné šifrovací algoritmy jako TrueCrypt, ale k tomu ještě navíc algoritmus Camellia a Kuznyechik. Dalším rozdílem je to, že VeraCrypt používá vyšší počty iterací u hešovacích algoritmů a podporuje dynamické nastavování jejich počtu. Činnost programu probíhá takovým způsobem, že umožňuje uživateli při vytváření šifrovaného diskového oddílu nastavit si počet iterací k hešovacímu algoritmu. Uživatelem nastavené číslo není konečný počet iterací, ale ten je vypočítán podle určitého vzorce [9].

FileVault je nástroj k šifrování disku. Používá šifrování XTS-AES 128 a je vyvinutý pro operační systém OS X [1].

CipherShed je šifrovací nástroj založený na TrueCryptu. Podporuje Windows, Linux i OS X [5]. Vzhledem k tomu, že CipherShed používá stejné šifrovací algoritmy a má stejnou strukturu hlavičky, jako TrueCrypt, je možné ho dešifrovat stejným způsobem jako diskové oddíly šifrované nástrojem TrueCrypt.

Kapitola 5

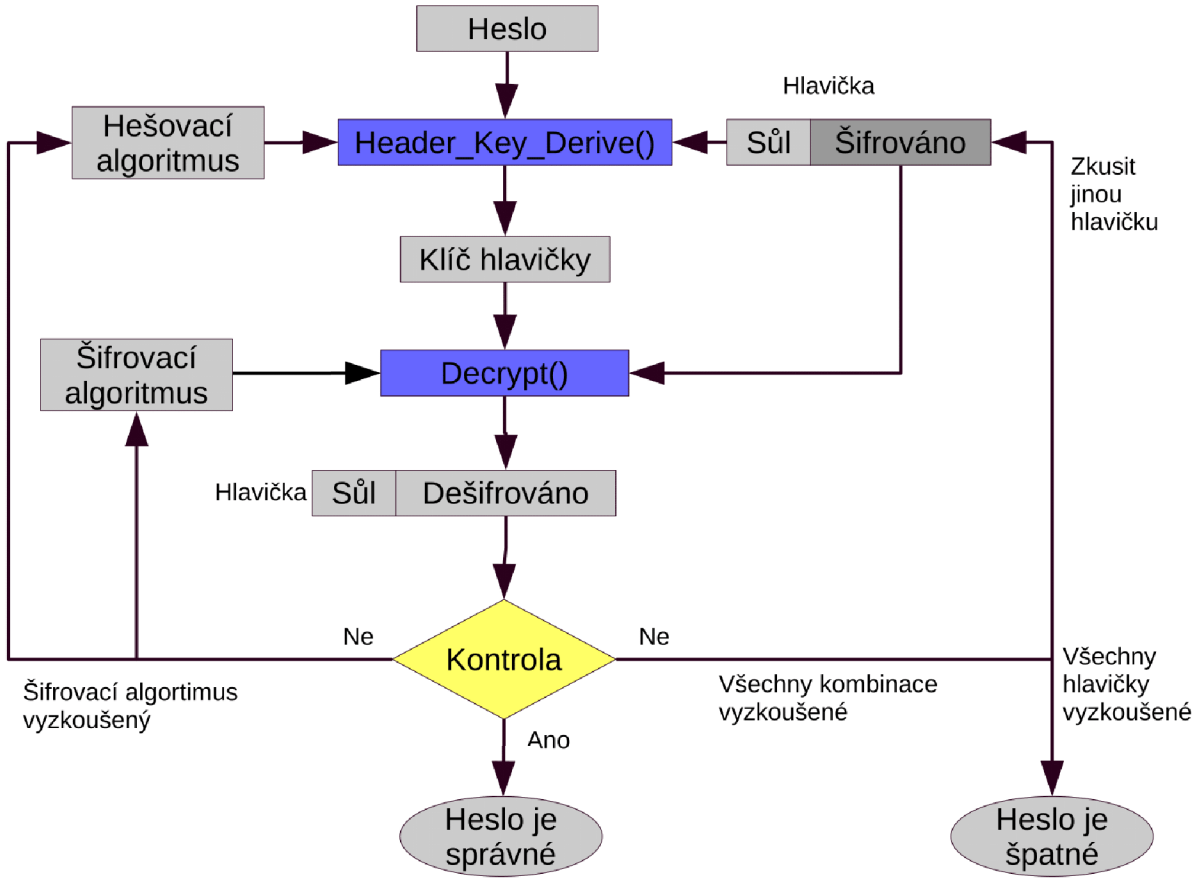
Návrh modulu

Výsledný modul bude provádět obnovu hesel zašifrovaného svazku TrueCryptu. Heslo k dešifrování svazku je uloženo v hlavičce. Neboť šifrovaný svazek TrueCryptu může obsahovat až 4 hlavičky, je nutné stanovit, kterou se bude modul pokoušet dešifrovat. Všechny hlavičky mají stejnou strukturu, jediný rozdíl je v tom, kde jsou uložené. Hlavička svazku TrueCryptu je zašifrovaná, jedinou nezašifrovanou částí hlavičky je prvních 64 bytů, ve kterých je uložena kryptografická sůl. K dešifrování hlavičky je zapotřebí kromě této soli také heslo, které se získá z generátoru Fitcracku. Vzhledem k tomu, že TrueCrypt nemá nikde uloženou informaci o tom, jakým způsobem byla hlavička svazku šifrovaná, je nutné vyzkoušet všechny možné způsoby dešifrování. TrueCrypt používá 8 šifrovacích algoritmů nebo jejich konkatenací a 3 hešovací algoritmy, takže možností je celkem $8 \times 3 = 24$. Výchozí je hešovací algoritmus RIPEMD-160 a šifrovací algoritmus AES v režimu XTS. K dešifrování potom stačí jen ověřit správnost hesla, z dešifrované hlavičky zjistit jakými algoritmy je šifrovaná datová část, vyjmout z ní klíč a dešifrovat datovou část. Tento modul má tedy za úkol dešifrovat hlavičku a ověřit, že použité heslo bylo správné.

Proces dešifrování hlavičky začíná tím, že se zavolá funkce pro odvození klíče (PBKDF2), která pomocí zkoušeného hešovacího algoritmu odvodí klíč hlavičky [21]. Jak už bylo zmíněno, modul musí zkoušet všechny tři hešovací algoritmy. Pokaždé se odvodí 192 bytů. Jedná se právě o 192 bytů, protože se s použitým hešovacím algoritmem musí vyzkoušet všechny kombinace šifrovacích algoritmů a 192 bytů má klíč s maximální možnou velikostí. Z vypočítaných 192 bytů se vyjme potřebný primární klíč a sekundární klíč (kvůli režimu XTS) pro každou kombinaci šifrovacích algoritmů. Následně se postupně zkouší osm daných kombinací šifrovacích algoritmů k dešifrování zbytku hlavičky (byty 64-511 při indexování od 0). A následně se zkontroluje se zda byty dešifrované hlavičky odpovídají tomuto:

- head[64-67] = "TRUE",
- head[72-75] = CRC32(head[256-511]),
- head[252-255] = CRC32(head[64-251]).

Přičemž head představuje hlavičku o 512 bytech. Indexování se počítá od 0. Pokud všechny rovnice odpovídají, potom to znamená, že heslo a algoritmy byly správně zvolené. V případě, že to tak není, je nutné zkoušet dané kombinace šifrovacích algoritmů a hešovacích algoritmů dál. Nastane-li situace, že jsou vyčerpány všechny kombinace algoritmů, musí se vyzkoušet další heslo. Princip obnovy hesla zašifrovaného svazku TrueCryptu je znázorněn na obrázku 5.1 [20, 21].



Obrázek 5.1: Princip obnovy hesla

Dále je možné ještě zkoušet obnovovat hesla hlavičky skrytého svazku (pokud existuje), která má stejnou strukturu, jako hlavička normálního svazku, ale je uložena s ofsetem 65536. V případě podezření, že výše zmíněné hlavičky mohou být poškozené, je možné zkoušet dešifrovat ještě jejich zálohy. Záložní hlavičky jsou šifrovány stejnými hesly jako ty hlavičky, které zálohují. Rozdíl však spočívá v kryptografické soli, která je v záložních hlavičkách jiná, než v těch původních.

Kapitola 6

Implementace

V této kapitole je popsán způsob implementace nového modulu Ficracku. Rozebírá se zde rozšiřující třída `TrueCrypt` nástroje FitXtractor, ve Fitcracku je vytvořena nová stejnojmenná třída, dále byly doplněny některé řídicí argumenty do třídy `Settings`. Do Fitcracku musely být doplněny i implementace kryptografických algoritmů, které nový modul používá.

6.1 FitXtractor

FitXtractor je nástroj, který slouží k extrakci dat ze souborů, jejichž hesla jsou obnovována pomocí Fitcracku. V případě TrueCryptu se může jednat i o diskové oddíly.

Ve FitXtractoru byla vytvořena třída `TrueCrypt` rozšiřující třídu `XtractorFile`, ve třídě `TrueCrypt` je implementováno čtení informací z hlaviček TrueCryptu, které jsou potřeba k obnově hesel. Jeho výstupem je soubor formátu XML, kde jsou uloženy informace ze všech 4 hlaviček. Nástroj Fitcrack následně dostává tento soubor jako svůj vstup.

6.2 Fitcrack

Třída `TrueCrypt`, která rozšifruje třídu `Cracker`, je řídicím jádrem nového modulu. Prostřednictvím této třídy probíhá proces obnovy hesel. Komunikace nástroje Ficrack s novým modulem probíhá prostřednictvím těchto hlavních metod: `checkPassword`, `verifyPassword` a `assignGPU`. Dále jsou implementovány další pomocné metody.

Konstruktor této třídy dostane prostřednictvím argumentů veškeré informace potřebné k obnově hesel, které byly přečteny ze zdrojového souboru XML pomocí nástroje FitXtractor. Potom je zavolána metoda `parseSettings`, která zkontroluje explicitně nastavené vstupní parametry, jako například použití jednoho hešovacího algoritmu.

`checkPassword` je metoda, jenž vyzkouší odvození klíčů a dešifrování hlavičky pomocí všech kombinací šifrovacích a hešovacích algoritmů, které TrueCrypt v současnosti používá. Nejdříve se ze vstupního hesla odvodí klíč pomocí metody `deriveKey`. Získaný odvozený klíč použije k inicializaci šifrovacích algoritmů pomocí metody `setXTSKeys`. K dešifrování hlavičky pomocí odvozeného klíče se použije funkce `DecryptBufferXTS`. Zmíněná funkce je převzatá ze zdrojových kódů nástroje TrueCrypt. Dešifrovaná hlavička se ověří pomocí metody `checkResult`.

Metoda `verifyPassword` nejdříve prověří, zda byla hlavička úspěšně dešifrována pomocí metody `checkResult`. Pokud se ověření hlavičky zdaří, tak se v případě režimu "verbose"vypíše informace o dešifrované hlavičce.

`assignGPU` je metoda, která slouží k inicializaci OpenCL kernelu `truecrypt`, jehož každá instance provádí ověření správnosti právě jednoho hesla. Instance kernelu `truecrypt` jsou spouštěny paralelně na GPU. Ověřování hesla na kernelu probíhá prakticky stejným způsobem jako za pomocí metody `checkPassword` na CPU. Kernel `truecrypt` má 7 parametrů. Parametry `password_array`, `pass_max_len`, `found_flag` a `found_vector` slouží k přenosu zkoušených hesel a značení, zda a kterou instancí kernelu bylo nalezeno správné heslo. Parameter `encrypted_header` je hlavička určená k dešifrování a poslední dva parametry `const_only_hash` a `const_only_cipher` slouží ke specifikaci, které algoritmy se mají zkoušet k dešifrování hlavičky.

Dalšími podstatnými metodami jsou `checkResult`, `setXTSKeys` a `deriveKey`. `checkResult` slouží k ověření správnosti dešifrování hlavičky, k tomu využívá funkci `GetCrc32`. `setXTSKeys` se používá k inicializaci šifrovacích algoritmů pro dešifrování v režimu XTS (vizte sekci 3.5.1). Metoda `deriveKey` slouží k samotnému odvození klíče.

Vzhledem k tomu, že všechny šifrovací algoritmy pracují v režimu XTS, stává se řídicí funkcí šifrování funkce `DecryptBufferXTS`, která vykonává dešifrování v režimu XTS. Zmíněná funkce pracuje s již inicializovanými strukturami pro šifrovací algoritmy.

Implementace kryptografických algoritmů byly převzaty ze zdrojových kódů nástroje TrueCrypt. Některé z převzatých funkcí musely být upraveny pro třídu `TrueCrypt` a pro kernel v OpenCL. V implementaci kernelu pomocí OpenCL jsou použity další veřejně dostupné implementace algoritmů SHA-512, Ripemd160 a AES (vizte kapitolu 3).

Kapitola 7

Experimenty

Tato kapitola se zabývá experimentováním s různými kombinacemi kryptografických algoritmů a různým nastavením Fitcracku. Měření je rozděleno do několika částí. Nejdříve provedeme experiment, kdy zkoušíme všechny kombinace hešovacích a šifrovacích algoritmů. Druhý experiment je zaměřen na obnovu hesla se zaměřením na rozlišení rychlosti různých hešovacích algoritmů. Třetí experiment se zabývá případem, kdy zkoušíme 1 hešovací a k němu všechny kombinace šifrovacích algoritmů, abychom zjistili, jak je rychlost obnovy hesel ovlivněna šifrovacími algoritmy. Poslední experiment se zabývá srovnáním Fitcracku s jiným řešením obnovy hesel šifrovaných diskových oddílů TrueCryptu.

K experimentování použijeme počítač s procesorem Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz a grafickou kartou AMD R9 Fury X. K obnově hesel byla zvolena znaková sada, která obsahuje jen malá písmena abecedy v kódování ASCII. Hesla byla zvolena tak, aby byla generována jako poslední (nejhorší) případ (z, zz, zzz, ...). Vzhledem k časové náročnosti obnovy hesel některá měření musela být aproximována. K výpočtu časů potřebných k obnově dlouhých hesel byl použit skript Ing. Radka Hranického.

7.1 Všechny algoritmy

Tato sekce se zabývá pravděpodobně nejběžnějším případem obnovy hesel. Je to případ, kdy uživatel Fitcracku nezná kryptografické algoritmy, které byly použity k šifrování hlavička šifrovaného diskového oddílu nástroje TrueCrypt. V tabulce 7.1 jsou uvedeny výsledky měření obnovy hesel na CPU a v tabulce 7.2 naměřená data z experimentování na GPU. Šifrovaný diskový oddíl, který byl zvolen k měření byl šifrován hešovacím algoritmem SHA-512 a touto kombinací šifrovacích algoritmů: Twofish-Serpent. Kombinace kryptografických algoritmů byla takto vybrána právě proto, že se jedná o poslední kombinaci, kterou nový modul zkouší při obnově hesla. Z výsledků experimentů vyplývá, že použití 4 vláken procesoru obnovu hesel urychlí 3,85krát. Daleko větší rychlosti lze dosáhnout za použití GPU, kdy se rychlost zvýší 8,51x oproti plnému výkonu procesoru (4 vlákna).

Délka hesla	1	2	3	4	5	6	7	Rychlost
1 vlákno	1s	27s	11m 40s	*5h 4m*	*5d 12h*	*143d*	*10y 68d*	26p/s
4 vlákna	2s	8s	3m 2s	*1h 19m*	*1d 10h*	*37d 4h*	*2y 236d*	100p/s

Tabulka 7.1: Výsledky experimentu při zkoušení všech algoritmů na CPU

Délka hesla	1	2	3	4	5	6	7	Rychlost
GPU	22s	23s	41s	9m 18s	*4h 2m*	*4d 8h*	*113d 14h*	851p/s

Tabulka 7.2: Výsledky experimentu při zkoušení všech algoritmů na GPU

	Délka hesla	2	3	4	5	6	7	Rychlost
Heš. alg.	Ripemd160	5s	1m 43s	44m 30s	*19h 16m*	*20d 21h*	*1y 179d*	178p/s
	SHA-512	2s	30s	13m 3s	*5h 39m*	*6d 3h*	*159d 6h*	607p/s
	Whirlpool	3s	50s	21m 24s	*9h 16m*	*10d 1h*	*261d 7h*	370p/s

Tabulka 7.3: Hešovací algoritmy za použití všech kombinací šifrovacích algoritmů na CPU (4 vlákna)

7.2 Hešovací algoritmy

Tento experiment je zaměřen na srovnání výpočetní náročnosti jednotlivých hešovacích algoritmů. Hlavička diskového oddílu, která byla při tomto experimentu dešifrována, měla pokaždé stejnou kombinaci šifrovacích algoritmů. Byla zvolena kombinace Twofish-Serpent, která se při obnově hesel zkouší jako poslední (nejhorší možnost). Při všech těchto experimentech byl pevně stanoven hešovací algoritmus a zkoušely se všechny kombinace šifrovacích algoritmů. Naměřená a vypočítaná data z experimentování na CPU jsou uvedena v tabulce 7.3. Výsledky z experimentu na GPU jsou v tabulce 7.4. Výpočetní náročnost jednotlivých hešovacích algoritmů se velmi liší. Pomocí SHA-512 lze odvodit klíč na CPU více než 3krát rychleji, než v případě použití Ripemd160. Na GPU nastává úplně jiná situace, zmíněný algoritmus se díky nevyrovnaným projevům akcelerace stává nejrychleji vypočítatelným ze všech hešovacích algoritmů (vizte tabulku 7.4).

7.3 Šifrovací algoritmy

V tomto experimentu byl k šifrovacím algoritmům zvolen hešovací algoritmus SHA-512, protože je ze všech hešovacích algoritmů nejméně výpočetně náročný a díky tomu se ve výsledném čase více projevují šifrovací algoritmy. Výpočet tedy probíhá tak, že se při obnově hesel z každého hesla odvodíme jeden klíč, pomocí algoritmu SHA-512, a následně ho použijeme pro jednu zvolenou kombinací šifrovacích algoritmů. Výsledky experimentu na CPU jsou uvedeny v tabulce 7.5. Z výsledků lze usoudit, že šifrovací algoritmy nemají na rychlost výpočtu ve srovnání s hešovacími téměř žádný vliv. Rozdíl v rychlosti se objevuje jen mezi výpočtem všech algoritmů (vizte sekci 7.1) a naměřenými hodnotami v tabulce 7.5.

	Délka hesla	1	2	3	4	5	6	7	Rychlost
Heš. alg.	Ripemd160	9s	9s	14s	2m 35s	*1h 7m*	*1d 5h*	*31d 12h*	3066p/s
	SHA-512	9s	10s	15s	2m 43s	*1h 10m*	*1d 6h*	*33d 3h*	2915p/s
	Whirlpool	13s	13s	22s	4m 28s	*1h 56m*	*2d 2h*	*54d 12h*	1773p/s

Tabulka 7.4: Hešovací algoritmy za použití všech kombinací šifrovacích algoritmů na GPU

	Délka hesla	2	3	4	5	6	7	Rychlost
Šifr. alg.	AES	2s	30s	12m 52s	*5h 35m*	*6d 1h*	*157d 11h*	614p/s
	Serpent	2s	30s	12m 53s	*5h 35m*	*6d 1h*	*157d 11h*	614p/s
	Twofish	2s	30s	12m 52s	*5h 35m*	*6d 1h*	*157d 11h*	614p/s
	AES-Twofish	2s	30s	12m 52s	*5h 35m*	*6d 1h*	*157d 11h*	614p/s
	AES-Twofish-Serpent	2s	30s	12m 54s	*5h 35m*	*6d 1h*	*157d 11h*	614p/s
	Serpent-AES	2s	30s	12m 53s	*5h 35m*	*6d 1h*	*157d 11h*	614p/s
	Serpent-Twofish-AES	2s	30s	12m 53s	*5h 35m*	*6d 1h*	*157d 11h*	614p/s
	Twofish-Serpent	2s	30s	12m 53s	*5h 35m*	*6d 1h*	*157d 11h*	614p/s

Tabulka 7.5: Šifrovací algoritmy za použití SHA-512 na CPU (4 vlákna)

7.4 Shrnutí výsledků

Z výsledků jasně vyplývá, že výpočetně nejnáročnější jsou hešovací algoritmy. Mezi hešovacími algoritmy dochází k velkým rozdílům ve výpočetní náročnosti. Algoritmus Ripemd160 je výpočetně nejnáročnější (vizte tabulku 7.3), avšak ostatní algoritmy používají dvakrát méně iterací. Ripemd160 se počítá s 2000 iterací a ostatní s 1000 iterací. Úplně jiná situace nastává na GPU, kde se dosahuje u každého algoritmů jiné akcelerace a dříve nejpomalejší algoritmus se stává nejrychlejším i přesto, že má nejvíce iterací.

Měření zaměřené na rychlosti výpočtu různých šifrovacích algoritmů má takové výsledky, že mají na výpočet velmi nízký vliv. Vzájemně se neliší se totiž ani o 1 heslo za sekundu (vizte tabulku 7.5). Pro rozlišení, jak moc daný šifrovaný algoritmus ovlivňuje rychlost obnovy hesel by bylo nutné provézt velmi dlouhá měření. Rychlost obnovy hesel při použití jednoho šifrovacího algoritmu se liší od případu, kdy se zkouší všechny šifrovací algoritmy v případě použití SHA-512 o 7 hesel za sekundu (vizte rozdíl v rychlost v tabulce 7.5 a řádek zabývající se SHA-512 v tabulce 7.3). V praxi tedy pevné určení šifrovacího algoritmu nemá velký vliv na rychlost výpočtu.

Hodnoty v tabulce 7.1 vypovídají o tom, že výpočet na 1 vlákně procesoru je efektivnější, než na 4 vláknech, přičemž zrychlení je 3,85krát. Dále z tabulek 7.2 a 7.4 lze vyčíst, že výpočet akcelerovaný na GPU je 8,51krát rychlejší, než na CPU při použití 4 vláken.

Kapitola 8

Závěr

Nejdříve jsem zanalyzoval nástroj TrueCrypt a kryptografické a jím používané kryptografické algoritmy. Následně jsem navrhl nový modul k obnově hesel diskových oddílů šifrovaných tímto nástrojem. Dále jsem implementoval nový modul Fitcracku s možností běhu na CPU i GPU. Na závěr jsem provedl sadu experimentů, ve kterých jsem porovnal rychlosti obnovy hesel za použití různých kryptografických algoritmů a rychlost obnovy hesel na CPU oproti akcelerovanému výpočtu GPU.

Informace potřebné k dešifrování daného oddílu jsou uloženy v jeho hlavičce, z toho vyplývá, že k jeho dešifrování je nutné nejdříve dešifrovat jeho hlavičku. Ta může být šifrována více možnými způsoby, přičemž informace o použitých kryptografických algoritmech nejsou nikde uvedeny. Neznalost algoritmů, kterými byl svazek šifrován způsobuje celý proces obnovy hesel velmi výpočetně náročným, protože je nutné vyzkoušet k dešifrování hlavičky všechny kombinace kryptografických algoritmů, kterými mohla být zašifrována. TrueCrypt umožňuje použití 24 různých kombinací kryptografických algoritmů k šifrování hlavičky oddílu. Dále se uvnitř svazku může nacházet i druhý skrytý svazek, který může být šifrovaný jiným způsobem a pomocí svého vlastního hesla. Dalším problémem je to, že šifrovaný svazek nemá žádnou signaturu, díky čemuž se jeví pouze jako změť náhodných čísel, tudíž není možné rozpoznat zda se nepokoušíme dešifrovat nesmyslná data nebo hlavičku skrytého oddílu.

Nový modul Fitcracku funguje tak, že z každého hesla nejprve odvodí klíč hlavičky k čemuž využívá jeden z několika hešovacích algoritmů. Na základě odvozeného klíče zkouší hlavičku dešifrovat pomocí různých šifrovacích algoritmů a nakonec ověřuje správnost hesla. Jestliže se nepodaří úspěšně dešifrovat hlavičku, tak se z daného hesla odvodí klíč pomocí jiného hešovacího algoritmu a pokud se neúspěšně vyzkouší všechny hešovací algoritmy, tak se zkusí k dešifrování použít další heslo. Implicitně se provádí dešifrování pomocí všech kombinací kryptografických algoritmů, ale je možné explicitně si nastavit které algoritmy se mají k dešifrování použít. Výpočet může běžet v akcelerovaném režimu na GPU.

Případným rozšířením této práce by mohla být implementace podpory tzv. Keyfiles a dále například režimů blokových šifer CBC a LRW, které byly používány ve starších verzích TrueCryptu. Dalším rozšířením práce by mohla být podpora VeraCryptu, který se od TrueCryptu příliš neliší a samotná implementace by mohla být založena na implementaci modulu, který provádí obnovu hesel diskových oddílů šifrovaných TrueCryptem.

Literatura

[1] Apple Inc.: *FileVault*. 2017, [Online; navštíveno 16.01.2017].
URL https://support.apple.com/cs-cz/HT204837

[2] Babakhani, A.; Bell, S.; Bradley, K.: *Whirlpool Cryptographic Hash Algorithm*. 2013, [Online; navštíveno 16.01.2017].
URL https://www.cs.rit.edu/~ark/spring2013/482/team/u4/report.pdf

[3] Benvenuto, C. J.: Galois field in cryptography. 2012.

[4] Borrelli, C.: IEEE 802.3 cyclic redundancy check. 2001.

[5] CipherShed Project: *CipherShed User's Guide, version 0.7.3.0*. 2014.

[6] Czeskis, A.; Hilaire, D. J. S.; Koscher, K.; aj.: Defeating Encrypted and Deniable File Systems: TrueCrypt v5.1a and the Case of the Tattling OS and Applications.

[7] Dobbertin, H.; Bosselaers, A.; Preneel, B.: RIPEMD-160: A Strengthened Version of RIPEMD. 1996.

[8] Hranický, R.: *Fitcrack*. [Online; navštíveno 16.01.2017].
URL http://wrathion.fit.vutbr.cz/fitcrack

[9] IDRIX: *VeraCrypt User's Guide, version 1.19*. 2016.

[10] Kaliski, B.: *PKCS #5: Password-Based Cryptography Specification Version 2.0*. Technická Zpráva 2898, Září 2000.
URL http://www.rfc-base.org/txt/rfc-2898.txt

[11] Khronos Group: *Khronos Launches Heterogeneous Computing Initiative*. Technická zpráva, [Online; navštíveno 16.01.2017].
URL https://www.khronos.org/news/press/releases/khronos_launches_heterogeneous_computing_initiative

[12] Khronos OpenCL Working Group: *The OpenCL Specification*. Technická zpráva, 2017, [Online; navštíveno 16.01.2017].
URL https://www.khronos.org/registry/cl/specs/opencl-2.2.pdf

[13] Krawczyk, H.; Bellare, M.; Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. Technická Zpráva 2104, Únor 1997.
URL https://www.ietf.org/rfc/rfc2104.txt

[14] Kryptotel Fz LLC: Kryptotel - How works Serpent Algorithm. 2010, [Online; navštíveno 16.01.2017].
URL http://en.kryptotel.net/serpent.html

[15] Mendel, F.; Pramstaller, N.; Rechberger, C.; aj.: On the collision resistance of RIPEMD-160. 2006.

[16] National Institute of Standards and Technology: FIPS PUB 197: ADVANCED ENCRYPTION STANDARD (AES). 2001, [Online; navštíveno 16.01.2017].
URL http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[17] Rijmen, V.; Barreto, P.: The WHIRLPOOL hash function. *World-Wide Web document*, ročník 72, 2001.

[18] Schmied, J.: *GPU akcelerované prolamování šifer*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2013.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=16858

[19] Schneier, B.; Kelsey, J.; Whiting, D.; aj.: Twofish: A 128-Bit Block Cipher. 1998.

[20] TrueCrypt Foundation: *TrueCrypt User's Guide, version 7.1a*. 2012.

[21] Zhang, L.; Zhou, Y.; Fan, J.: The forensic analysis of encrypted Truecrypt volumes. IEEE, May 2014, ISBN 978-1-4799-2033-4, s. 405–409.

Přílohy

Příloha A

Obsah přiloženého paměťového média

V přiloženém paměťovém médiu se nachází tyto hlavní složky a soubory:

- README – seznam zdrojových souborů, které do Fitcracku byly přidány nebo upraveny, příklady testovacích příkazů,
- latex/ – zdrojové soubory technické zprávy,
- fitxtractor/ – zdrojové soubory nástroje FitXtractor,
- fitxtractor/README – soubor s informacemi, jak spustit nástroj FitXtractor,
- test/fitxtractor/ – soubory k testování rozšíření FitXtractoru,
- fitcrack/ – zdrojové soubory nástroje Fitcrack,
- fitcrack/README – soubor s informacemi, jak spustit nástroj Fitcrack,
- test/fitcrack/ – soubory k testování rozšíření Fitcracku,
- projekt.pdf – technická zpráva.