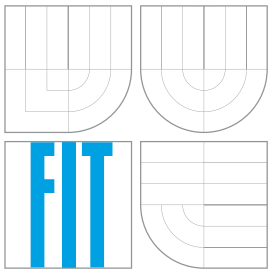
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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

OBNOVA HESEL DOKUMENTŮ MICROSOFT OFFICE S VYUŽITÍM GPU

MICROSOFT OFFICE PASSWORD RECOVERY USING GPU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ ZOBAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK HRANICKÝ

BRNO 2016

Zadání bakalářské práce/18341/2015/xzobal00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Zobal Lukáš**
Obor: Informační technologie
Téma: **Obnova hesel dokumentů Microsoft Office s využitím GPU**
Microsoft Office Password Recovery Using GPU
Kategorie: Bezpečnost

Pokyny:

1. Seznamte se s architekturou a implementací nástroje Wrathion.
2. Nastudujte techniky šifrování dokumentů Microsoft Office.
3. Navrhněte rozšíření modulů programu, které zajistí podporu dalších verzí dokumentů Microsoft Office.
4. Navržené rozšíření implementujte (včetně kódu pro akceleraci pomocí GPU).
5. Proveďte měření výkonu navržených modulů a výsledky porovnejte s konkurenčními nástroji.
6. Zhodnoťte dosažené výsledky.

Literatura:

- WU, Xiaohong, HONG, Jingxin, ZHANG, Yixiong. Analysis of OpenXML-based office encryption mechanism. In: *7th International Conference on Computer Science & Education (ICCSE)*. Melbourne: IEEE 2012. s. 521-524. ISBN 978-1-4673-0241-8.
- 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 2015
Datum odevzdání: 18. května 2016

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

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá obnovou hesel dokumentů Microsoft Office rozšířením existujícího nástroje Wrathion. V práci je rozebrána problematika zabezpečení dokumentů, dále moderní metody šifrování a hešování a také základy standardu OpenCL. Je provedena analýza struktury formátů MS Word, MS Excel a MS PowerPoint všech verzí od roku 1997. Na základě těchto znalostí je pak navrženo a implementováno jednak rozšíření existujícího modulu DOC o novější verze šifrování, jednak zcela nové moduly pro formáty XLS a PPT a také jejich novější varianty DOCX, XLSX a PPTX. S implementovanými moduly je následně provedeno měření výkonu a porovnání s konkurenčními nástroji v oblasti obnovy hesel.

Abstract

This thesis describes the password recovery of Microsoft Office documents by expanding an existing tool Wrathion. The thesis explains the issue of digital document protection, modern encryption and hashing algorithms and rudiments of OpenCL standard. Next, the analysis of structure of MS Word, MS Excel and MS PowerPoint documents is performed, including all the versions since 1997. Using this knowledge, we create a draft and an implementation of improved DOC module for newer versions of the encryption, as well as a draft and an implementation of brand new modules for XLS and PPT formats and their newer variants DOCX, XLSX and PPTX. After that, we measure performance of the new modules and compare it with other competing password recovery tools.

Klíčová slova

kryptografie, obnova hesel, Wrathion, Microsoft Office, Office Open XML, DOC, DOCX, XLS, XLSX, PPT, PPTX, OpenCL, GPU

Keywords

cryptography, password recovery, Wrathion, Microsoft Office, Office Open XML, DOC, DOCX, XLS, XLSX, PPT, PPTX, OpenCL, GPU

Citace

ZOBAL, Lukáš. *Obnova hesel dokumentů Microsoft Office s využitím GPU*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Hranický Radek.

Obnova hesel dokumentů Microsoft Office s využitím GPU

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Hranického. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Zobal
12. května 2016

Poděkování

Tímto bych rád poděkoval Ing. Radku Hranickému za odbornou pomoc a vedení při vypracovávání této bakalářské práce.

Obsah

Kapitola 1

Úvod

Kryptografie vznikla před mnoha tisíci lety. Ať již ve formě rytin na hliněných tabulkách nebo posloupnosti bitů v síti, vždy šlo o zabezpečení informací. Zatímco do poloviny 20. století šlo spíše o práci s tužkou a papírem, s rozvojem výpočetní techniky, hlavně z důvodu druhé světové války, začaly vznikat šifry, které nejsou rozluštitelné pouze lidskou rukou. Spolu s vývojem kryptografie se přirozeně vyvíjela také kryptoanalýza, tedy schopnost rozšifrovat skrytá data bez znalosti klíče.

Dnešní šifrovací algoritmy již vyžadují k šifrování i rozšifrování dat výhradně počítače. Stejně tak počítačů využívají kryptoanalytici. Hlavní překážkou pro úspěšné prolomení šifry je výkon počítačů. Ten se však každým rokem značně zvyšuje. Z toho důvodu je použití šifer navržených před několika desítkami let považováno za riskantní a jsou navrhovány nové algoritmy, které by odolaly stále propracovanějším útokům.

Obnova hesel ve formě kryptoanalýzy má velký význam pro vyšetřovatele. Zjištění důležitých informací v co nejkratší době může hrát klíčovou roli pro vyřešení případu. Nástroj, který v rámci tohoto projektu rozšiřujeme, je určený právě pro tento účel. Jeho hlavní výhodou je schopnost urychlení této obnovy pomocí GPU. Tato zařízení totiž obsahují stovky až tisíce relativně samostatných výpočetních jednotek, které jsou schopny celý proces paralelizovat a významně tak urychlit.

Tato práce si dává za úkol rozšířit schopnosti nástroje Wrathion o další formáty dokumentů Microsoft Office, jmenovitě Microsoft Word, Excel a PowerPoint. Kapitola 2 se zabývá obecným způsobem zabezpečení dokumentů Microsoft Office. V kapitole 3 se podíváme na nástroj Wrathion, jeho architekturu, aktuální funkčnost a jak je možné ji rozšířit. V následující kapitole 4 je stručně popsán standard OpenCL, který je použit k paralelizaci celého procesu prolamování hesel. Podrobný rozbor struktury dokumentů Microsoft Office se nachází v kapitole 5. Návrh implementace jednotlivých modulů nalezneme v kapitole 6. Postup při implementaci modulů pro nástroj Wrathion je popsán v kapitole 7. V závěrečné kapitole 8 se podíváme na dosažené výsledky a srovnáme je s konkurenčními nástroji.

Kapitola 2

Způsob zabezpečení dokumentů

V této kapitole se podíváme na způsob, jakým je možné zabezpečit dokumenty Microsoft Office. Bude představena problematika hešovacích funkcí a detailněji rozebrány metody MD5, SHA-1 a SHA-2 (viz sekce 2.1.3, 2.1.4 a 2.1.5), které jsou v dokumentech Office použity. Také se seznámíme s funkčností kryptografických algoritmů a opět podrobněji probereme metody RC4 a AES (viz sekce 2.2.5 a 2.2.6), které jsou užity v rámci zašifrování dokumentů Office Word, Excel a PowerPoint.

2.1 Hešovací funkce

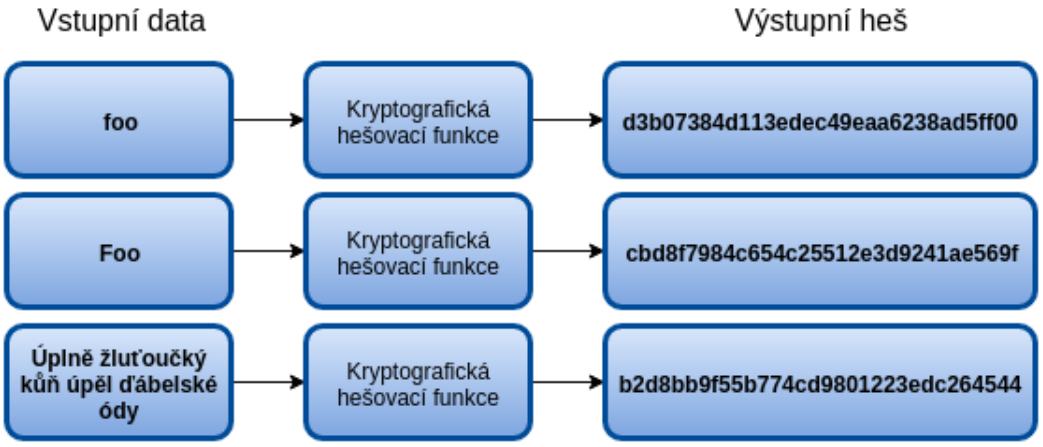
Hešovací funkce je matematická funkce, která dokáže namapovat data libovolné velikosti do datového řetězce fixní délky, většinou mnohem menší, než je délka vstupních dat. Takovému výstupu se říká výtah, otisk, či heš. Tyto funkce mají velké uplatnění nejen v informatice. Příkladem může být datový typ hešovací tabulka. Obecné vlastnosti hešovacích funkcí můžeme přehledně shrnout do následujících bodů:

- Heš lze jednoduše spočítat pro jakákoli vstupní data.
- Data o libovolné délce vyprodukují stejně dlouhý heš.
- Malou změnou na vstupu získáme velmi odlišný heš.

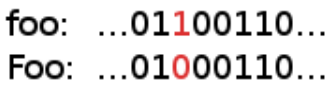
Pro naše potřeby však budeme uvažovat speciální varianty této funkce, tzv. kryptografické hešovací funkce. Takovéto funkce musejí splňovat několik důležitých předpokladů pro to, aby byly schopné zabezpečit citlivé informace. Kromě výše uvedených bodů tedy rozšíříme vlastnosti kryptografických hešovací funkcí o následující:

- Je prakticky nemožné získat z heše původní vstupní data.
- Je prakticky nemožné změnit data bez současné změny vyprodukovaného heše.
- Je prakticky nemožné nalézt dvě zprávy se stejným hešem.

Z těchto bodů plyne, že hešovací funkce jsou pouze jednosměrné. Je tedy velký rozdíl, zda jsou data zašifrována nebo zahešována. Při zašifrování dat můžeme použitím správného klíče získat původní data. Při hešování toto není možné.



Obrázek 2.1: Použití hešovací funkce na různých vstupech



Obrázek 2.2: Binární reprezentace vstupních řetězců

Pokud bychom se podívali na tuto problematiku z teoretického hlediska, zjistili bychom, že výše uvedené vlastnosti nelze splnit. Uvažujeme totiž fixní velikost výsledného heše a zároveň libovolný vstup. To nám dává nekonečné množství vstupů spolu s omezeným stavovým prostorem pro výslednou hešovací funkci. V praxi je však tento stavový prostor tak obrovský (například pro 128-bitovou funkci MD5 dostaneme přibližně $3,4 \cdot 10^{38}$ kombinací), že tento problém můžeme částečně ignorovat. Pokud dva vstupy produkují stejný heš, nazýváme to kolize. Čím jednodušeji lze takovou kolizi detekovat, tím méně je daný algoritmus užitečný.

Příklad použití hešovací funkce je možné vidět na obrázku 2.1. Je zde možné si všimnout například výše zmíněného pravidla, které říká, že jen nepatrná změna na vstupu vyprodukuje naprosto odlišný heš. Pokud bychom se podívali na binární data vstupních řetězců, liší se jen v jediném bitu (viz obrázek 2.2). Na předchozím obrázku však vidíme, že výsledné heše se nepodobají. Také je zde znázorněno, že vstupy odlišných délek generují fixní velikost heše.

2.1.1 Příklady použití

V praxi se hešování používá v různých oblastech. My se však zaměříme hlavně na oblast informatiky a bezpečnosti.

Velmi často se s hešovacími funkcemi setkáme při ověřování integrity příjimaných dat. Pro tyto data může existovat heš, který lze použít pro ověření, zda data, která jsme stáhli, jsou opravdu ta, která jsme si vyžádali a nebylo s nimi nijak manipulováno.

To souvisí také s digitálními podpisy. Heš vyprodukovaný dokumentem spolu s privátním klíčem autora podpisu zajistí autenticitu podpisu.

Pro nás nejdůležitější oblast, ve které se hešovací funkce používají, je hešování hesel. Ukládání hesel na jakékoli médium v jejich původní podobě je velmi nebezpečné, protože pokud se útočník k tomuto úložišti dostane, získá i přístup ke všem těmto heslům. Stejné riziko existuje při přenosu těchto hesel přes síť. Řešením je ukládat pouze heše těchto hesel.

Jak bylo zmíněno výše, kryptografické funkce jsou jednosměrné a tedy není možné z heše hesla získat původní hodnotu. Tu navíc nepotřebujeme ani při zadávání hesla uživatelem. Ze zadaného hesla se totiž pouze spočítá heš a ten se srovná s hešem uloženým v databázi. Pokud se heše shodují, je přístup povolen. Z toho vyplývá, že v některých případech ani není nutné znát správné heslo, pouze kombinaci, která vyprodukuje správný heš.

2.1.2 Používané metody

Heš, který byl použit v příkladu na obrázku 2.1 byl vytvořen pomocí hešovací funkce MD5. Ta je použita i ve verzi 97-2000 dokumentů Microsoft Office, proto se její funkcí budeme zabývat blíže v následující kapitole.

Dalšími používanými metodami je skupina kryptografických hešovacích funkcí SHA. Považuje se za nástupce MD5 a je použita v MS Office od verze 2003. SHA má několik verzí, které si popíšeme v následujících kapitolách.

2.1.3 MD5

Message Digest 5 (MD5) je rozšířená kryptografická hešovací funkce, která produkuje 128-bitový heš. Ten je typicky vyjádřen jako posloupnost 32 hexadecimálních číslic, jak je vidět na obrázku 2.1. Tato funkce byla vytvořena v roce 1991 a nahradila svého předchůdce MD4, která je již považována za prolomenou. Ani tato funkce však není dokonalá a byly v ní nalezeny první kolize [18]. Pro tento projekt je však důležitá, protože hesla dokumentů Microsoft Office verze 97-2000 jsou zahešovány právě touto funkcí.

Funkce MD5 funguje následovně [13]. Do vstupních dat je nejprve doplněna '1' a poté tolik '0', aby se celková délka vstupu po operaci modulo 512 rovnala 448. Tedy 64 bitů méně než je délka 512. Těchto 64 bitů se následně vyplní délkou dat (před přidáním bitů zmíněných výše). V tomto případě je porušena podmínka hešovací funkce, která říká, že můžeme vložit libovolně velký vstup. Jedná se ale pouze o teoretické porušení, protože velikosti 2^{64} bitů nemůže žádný soubor dosáhnout, alespoň ne v blízké budoucnosti.

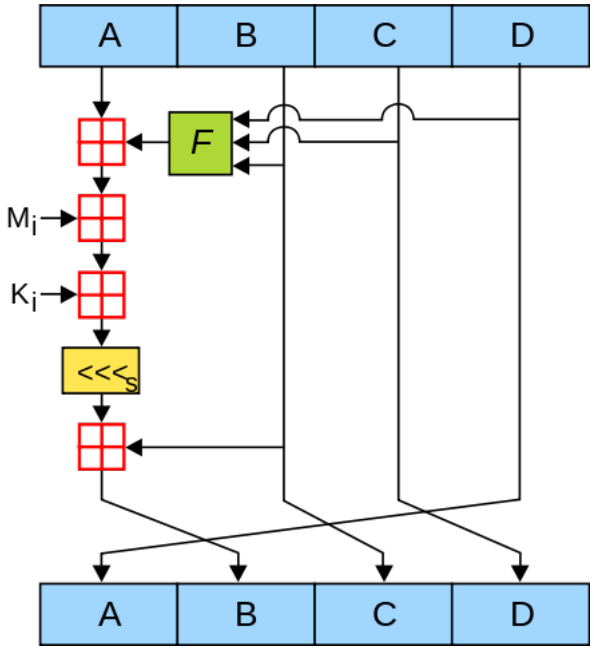
Poté jsou definovány čtyři 32-bitové registry A, B, C a D a jsou inicializováno předem zvolenou hodnotou. Dále jsou definovány 4 funkce F, G, H, I, které provádí různé bitové operace. Hlavní je, že jako vstup berou 32-bitovou hodnotu a také 32-bitovou hodnotu vrací jako výstup. 512-bitové bloky jsou poté rozděleny do šestnácti 32-bitových bloků. Ty jsou potom zpracovány ve čtyřech kolech, kde každé se opakuje 16x. To znamená 64 kol pro každý 512-bitový blok.

Jedno kolo této funkce je vidět na obrázku 2.3. M_i je jeden ze šestnácti 32-bitových bloků a K_i je konstanta, lišící se v každém kole.

2.1.4 SHA-1

Secure Hash Algorithm 1 (SHA-1) je kryptografická hešovací funkce, kterou navrhla v roce 1995 americká NSA[2] a vydal NIST[3] jako americký federální standard. Existuje několik různých verzí tohoto algoritmu. Nejrozšířenější byla verze SHA-1, která se velmi podobala MD5. Její předchůdce SHA-0, navržený v roce 1993 nebyl široce rozšířen a brzy byl nahrazen právě SHA-1, která eliminovala jeho slabiny. SHA-1 je použita ve verzi Office 2003 a také ve verzích 2007 a 2010, kde je pro zahešování klíče provedeno několik desítek tisíc cyklů

[1] Zdroj: https://en.wikipedia.org/wiki/MD5
[2] The National Security Agency – Národní bezpečnostní agentura v USA
[3] National Institute of Standards and Technology – Národní institut pro standardy v USA



Obrázek 2.3: Znázornění hešovací funkce MD5[1]

této hešovací funkce, kvůli ztížení obnovy hesel. Po objevených slabinách v roce 2005 [16] byla rozšířena novější verze – SHA-2.

SHA-1 produkuje 160-bitový heš, zapisovaný, stejně jako u MD5, v podobě hexadecimálních číslic – v tomto případě čtyřiceti. Funguje obdobně jako MD5 a nákres jednoho cyklu je vidět na obrázku 2.4. Písmena A–D opět znázorňují 32-bitový registry, F je nelineární funkce a $\lll_n$ je bitový posun.

2.1.5 SHA-2

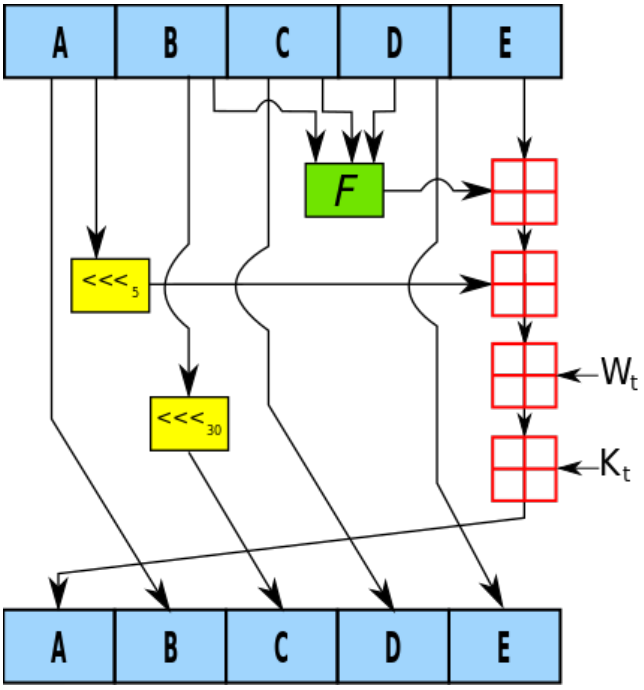
Secure Hash Algorithm 2 (SHA-2) je hešovací funkce, také známá jako SHA256 a SHA512. Byla vyvinuta již v roce 2001. Rozšířila se však až v roce 2005 s již zmíněným „prolomením“ svého předchůdce SHA-1. SHA-2 je použita v nejnovějších verzích MS Office, tedy 2013 a dále. Tato funkce má několik verzí, které produkují heše o délce 224, 256, 384 nebo 512 bitů, podle čehož se také funkce někdy nazývají (tedy například SHA512).

Nejnovější verze této rodiny je SHA-3 a ta se výrazně liší od předchozích verzí. Úřadem NIST byla však standardizována teprve v polovině roku 2015 a není zatím široce využívána. Detailněji se jí tedy věnovat nebudeme.

2.2 Šifrovací algoritmy

V této sekci se podíváme na to, co jsou to šifrovací algoritmy a na to, jak vznikly, k čemu se používají a jak se dají rozdělit. Poté si podrobně rozebereme šifry RC4 a AES, které se používají v dokumentech Office a budeme s nimi tedy přímo pracovat.

[4]Zdroj: https://en.wikipedia.org/wiki/SHA-1



Obrázek 2.4: Znázornění hešovací funkce SHA-1[4]

2.2.1 Historie kryptografie

Kryptografické algoritmy jsou vyvíjeny již od pradávna. První náznaky pokusů o zašifrování informací jsou tisíce let staré a pochází z dob starého Egypta[5]. Jedna ze známých šifer je Caesarova šifra [11]. Ta je o několik tisíc let mladší a jmenuje se po římském státníkovi Juliu Caesarovi, který ji používal v soukromé korespondenci. Jedná se o substituční šifru (viz níže).

Obecně platí, že šifrování je, a bylo, prováděno za účelem dodržení základních principů informační bezpečnosti.

- **Důvěrnost** – Obsah není k dispozici osobám, které k němu nemají mít přístup.
- **Autentizace** – Je zcela jasné, kdo data poslal.
- **Integrita** – Přijatá data nebyla nikým neoprávněným upravena.
- **Nepopiratelnost** – Autor nemůže popřít, že data opravdu vytvořil.

Pro všechny šifrovací funkce platí, že jsou obousměrné. To znamená, že při znalosti správného klíče můžeme získat původní obsah.

V době, kdy neexistovala výpočetní technika, se používaly šifrovací algoritmy, které můžeme rozdělit do několika skupin.

- **Steganografické šifry** – Tyto šifry obecně ukládají šifrovaný text uvnitř jiného textu. V praxi se tak mohlo jednat například o poslední slova každé věty. Nevýhoda tohoto postupu je, že jakmile se prozradí metoda šifrování, nedá se již použít. Dnes se tato metoda používá v rámci jednotlivých bitů určitého souboru, například bity

[5]Zdroj: http://www.cypher.com.au/crypto_history.htm

Obrázek 2.5: Obrázek, ve kterém je ukryta tajná zpráva

na prvočíselných indexech obrázku. Při pohledu na takový obrázek nelze nic poznat, viz obrázek 2.5. Hlavní nevýhoda však vyplývá z velikosti zašifrované zprávy, která je obvykle mnohonásobně větší, než původní data.

- **Substituční šifry** – Principem těchto šifer je nahrazení původních znaků jinými podle zvoleného klíče. Výhodou takto zašifrovaných souborů je obdobná velikost, jakou měla původní data. Nevýhoda je možnost použití frekvenční analýzy k prolomení této šifry, tedy znalosti o frekvenci výskytu jednotlivých písmen, digramů či trigramů v daném jazyce. Tomu se předchází vytvářením speciálních znaků pro tyto konstrukce, či nejčastější slova nebo použitím více metod substituce (tzv. polyalfabetická substituce).

- **Transpoziční šifry** – Takto šifrovaný text je tvořen stejnými znaky, jako původní data, pouze jsou tyto znaky v jiném pořadí. Příkladem může být šifra Rail Fence nebo sloupcová transpozice [19].

V moderní kryptografii se také používá kombinace těchto metod. Nejčastěji se jedná o kombinaci substituční a transpoziční šifry.

2.2.2 Typy útoků

Transformace šifrované informace do původní podoby bez znalosti klíče se nazývá kryptoanalýza. Obnova hesel, kterou se tento projekt zabývá, využívá právě kryptoanalýzy.

Útoky lze rozdělit podle množství útočníkovi poskytnutých informací na následující [14].

- **Ciphertext-only** – Útočník má k dispozici pouze kolekci zašifrovaných textů.

- **Known-plaintext** – K dispozici je sada zašifrovaných textů a k nim odpovídající původní data.

- **Chosen-plaintext/ciphertext** – Útočník si volí data, která jsou zašifrována neznámým klíčem, respektive zašifrovaný text, ke kterému nezná původní data.

- **Adaptive chosen-plaintext** – Je podobný předchozímu typu, útočník se ovšem může rozhodovat na základě získaných znalostí předchozích analýz.

- **Related-key attack** – Útočník má k dispozici šifry vytvořené dvěma klíči. Tyto klíče jsou neznámé, je však znám jejich vzájemný vztah.

Dále je útoky možné charakterizovat podle času, který je potřeba na prolomení šifry, tedy počet vykonaných kroků výpočetní jednotky. To bývá u útoků největší problém, hlavně při brute-force útocích. Další záležitostí, kterou se musí kryptoanalytik zabývat, je množství paměti, která je pro obnovu potřeba a množství zašifrovaných, respektive rozšifrovaných textů, které musí mít k dispozici.

Výsledky útoků mohou být různé. Útočník se může dozvědět kompletní obsah zprávy spolu s tajným klíčem. Nalezení obdobného algoritmu pro dešifrování zprávy, nebo jen objevení jednotlivých předem neznámých rozšifrovaných dokumentů vetší rychlostí, než je rychlost brute-force útoku, je považováno za prolomení šifrovacího algoritmu.

2.2.3 Moderní kryptografie

Od období druhé světové války došlo k obrovskému rozvoji kryptografie. Dnešní šifry již nelze v naprosté většině rozšifrovat jen pomocí tužky a papíru. Je třeba použití výpočetní síly počítačů.

Dnešní šifrovací algoritmy můžeme rozdělit na dvě skupiny [17].

- **Symetrické algoritmy** – Pro tyto algoritmy platí, že šifrování i rozšifrování probíhá pomocí jednoho tajného klíče. Pokud někdo zná tento klíč, může šifrovat i rozšifrovávat všechny zprávy, které používají tento klíč. Problém nastává při předávání těchto klíčů přes internet nebo jinou velkou síť, kde existuje nebezpečí prozrazení klíče nepovolané osobě.

- **Asymetrické algoritmy** – Tento problém řeší právě asymetrické šifrování. V tomto případě se používá dvojice klíčů, jeden tajný klíč, tzv. privátní, a jeden veřejný klíč. Data zašifrovaná veřejným klíčem lze rozšifrovat pouze privátním klíčem a opačně.

Za symetrické lze považovat i algoritmy používané v historii. Jejich výhodou je jednoduchost a rychlost šifrování. Naopak asymetrické algoritmy pro dostatečné zabezpečení vyžadují délku klíčů v řádu tisíců bitů a to se také projeví na času, který je potřebný pro zašifrování dat.

Další možné rozdělení šifrovacích algoritmů je na *blokové šifry* a *proudové šifry*.

- **Blokové šifry** – Tyto šifry pracují vždy s celým blokem dat (např 64 bitů). Rozlišujeme několik různých variant. Příkladem blokové šifry je například algoritmus DES nebo AES.

- **Proudové šifry** – Tyto algoritmy šifrují data bit po bitu, pomocí tzv. keystreamu[6]. Tyto bity jsou potom zkombinovány s bity původních dat, čímž vzniká zašifrovaný text. Většinou se jedná o operaci exklusivního součtu (tzv. XOR). Příkladem proudové šifry je RC4.

[6] Posloupnost náhodných bitů generovaná pomocí klíče

2.2.4 Režimy blokových šifer

Existuje několik způsobů, jak bloky dat opakovaně transformovat do zašifrovaných bloků, aby byly dodrženy všechny principy bezpečnosti. Podrobněji se budeme zabývat následujícími základními režimy.

- **Electronic Code Book (ECB)** – Jedná se o nejjednoudšší režim, kde každý blok je šifrován samostatně, bez použití inicializačního vektoru (dále IV). Je tedy bezpečný pouze pro zašifrování jediného bloku. Totožné původní data totiž vytvoří totožný výsledek.
- **Cipher Block Chaining (CBC)** – V tomto režimu je použit IV pro operaci exklusivní součet s prvním blokem dat, pro dosáhnutí náhodného výsledku i s totožnými daty. Navíc je nad každým dalším blokem dat proveden exklusivní součet s předešlým výsledkem šifrování. Výsledek šifrování tedy záleží na všech dosud zašifrovaných datových blocích.
- **Cipher Feedback (CFB)** – Metoda velmi podobná CBC. Na začátku je zašifrován IV a výsledek exklusivně sečten s blokem dat, čímž je získán výsledek. Tento výsledek je pro další blok zašifrován a výsledek exklusivně sečten s dalším blokem dat. To se opakuje pro všechny následující bloky.
- **Output Feedback (OFB)** – Metoda velmi podobná CFB, pouze se jako vstup šifrování použije ihned výsledek předchozího šifrování, bez aplikace exklusivního součtu.

Nyní se podíváme na příklady šifrovacích algoritmů.

2.2.5 RC4

Rivest Cipher 4 (RC4), pojmenovaná podle svého autora, je proudová šifra. Byla vynalezena již v roce 1987 Ronaldem Rivestem. Je velmi jednoduchá a rychlá, bohužel se však ukázalo, že je velmi zranitelná a náchylná k množstvím útoků. Velké firmy jako Microsoft nebo Mozilla její užití považují za nebezpečné a dokonce bylo vydáno RFC doporučení ohledně zákazu používání RC4 v protokolu TLS [12]. Tento šifrovací algoritmus je však použit ve starších verzích dokumentů Office a popíšeme si tedy, jak funguje.

Šifrování i dešifrování se provádí bit po bitu pomocí operace XOR původních dat s vygenerovaným keystreamem, respektive obráceně. Pro generování keystreamu je potřeba pole obsahující permutaci 256 prvků a dva 8-bitové ukazatele. Nejprve proběhne algoritmus *key-scheduling*, který iniciuje 256-prvkové pole. Ten je vidět v algoritmu 1. Jako vstup dostane klíč a jeho délku v bytech.

Vstup: klíč, délkaKlíče
Výstup: S
for *i* ***from*** *0* ***to*** *255* **do**
  S[i] = i;
end
j = 0;
for *i* ***from*** *0* ***to*** *255* **do**
  j = (j + S[i] + klíč[i % délkaKlíče]) % 256;
  swap S[i] **and** S[j];
end

Algoritmus 1: Key-scheduling algoritmus

Po inicializaci proběhne algoritmus *keystream-generator*, který upraví obsah pole podle zadaného klíče. Ze stavu pole se po skončení tohoto algoritmu generuje keystream. Pseudokód se nachází v algoritmu 2.

Vstup: S
Výstup: K
i = 0;
j = 0;
while *generování keystreamu* **do**
    i = (i + 1) % 256;
    j = (j + S[i]) % 256;
    swap S[i] **and** S[j];
    z = (S[i] + S[j]) % 256;
    K = S[z];
    output K;
end

Algoritmus 2: Algoritmus pseudo-náhodného generátoru

2.2.6 AES

Advanced Encryption Standard (AES) je šifra, která byla institutem NIST prohlášena v roce 2000 jako standard. Nahradila tak svého předchůdce z roku 1979, DES (*Data Encryption Standard*), respektive 3DES, které byly již považovány za nedostatečné a jednoduše prolomitelné. Zajímavé je, že AES byl vybrán na základě „veřejného“ procesu, ve kterém bylo množství kandidátů. Nakonec byl pro tento nový standard zvolen algoritmus *Rijndael*, navržený dvěma belgickými kryptografy. Jedná se o blokovou šifru. Velikost jednoho bloku je jednotná, 128 bitů, existují však tři varianty délku klíčů – 128, 192 a 256 bitů. Transformace na šifrovaná data probíhá v rámci několika cyklů. Počet těchto cyklů je dán délkou klíče – 10 cyklů pro 128-bitový klíč, 12 cyklů pro 192-bitový klíč a 14 cyklů pro 256-bitový klíč. AES operuje nad maticí bytů, například pro 128-bitový klíč je to matice 4x4 byty.

Samotný cyklus se pak skládá z několika kroků. Tyto kroky mohou obsahovat následující fáze.

- **Fáze SubBytes** – Každý byte je nahrazen svým záznamem ve fixní look-up tabulce.
- **Fáze ShiftRows** – Všechny řádky tabulky jsou posunuty o různý počet pozic vlevo.
- **Fáze MixColumns** – Každý sloupec je násobený fixní hodnotou.
- **Fáze AddRoundKey** – Provádí se exklusivní disjunkce pro každý byte.

Všechny tyto kroky se pak skládají do posloupnosti, která je vidět v algoritmu 3.

Vstup: data, klíč
Výstup: šifrovaný text
podKlíče = keySchedule(klíč);
upravenáData = data;
for *i* ***from*** *0* ***to*** *k* **do**
    šifrovanáData = AddRoundKey(upravenáData);
    subData = SubBytes(šifrovanáData);
    shiftData = ShiftRows(subData);
    upravenáData = MixColumns(shiftData);
end
šifrovanáData = AddRoundKey(upravenáData);
subData = SubBytes(šifrovanáData);
výslednáŠifra = ShiftRows(subData);
return výslednáŠifra;

Algoritmus 3: Algoritmus pseudo-náhodného generátoru

Tato metoda je dnes považována za bezpečnou. Jsou známy postupy, kterými lze šifra teoreticky prolomit rychleji, než pomocí brute-force útoku. Žádný z nich však není na dnešních výpočetních strojích prakticky proveditelný [7].

Kapitola 3

Nástroj Wrathion

Wrathion je nástroj pro obnovu hesel dokumentů [10]. Vznikl jako součást diplomové práce pana Ing. Jana Schmieda v roce 2014 [15]. Jedná se o GPU akcelerované prolamování hesel s velkým důrazem na paralelní zpracování. To je dosaženo použitím standardu OpenCL.

3.1 Architektura nástroje

Nástroj Wrathion je navržen do následujících částí:

- **Jádro** – Obsahuje základní funkčnost potřebnou pro prolamování hesel. To zahrnuje řídící jednotku a generátory hesel.
- **Moduly** – Obsahují kód pro zpracování jednotlivých formátů. Všechny moduly jsou schopny pracovat jak s GPU tak CPU.
- **Aplikace** – Zapouzdřuje všechno do konzolové aplikace.

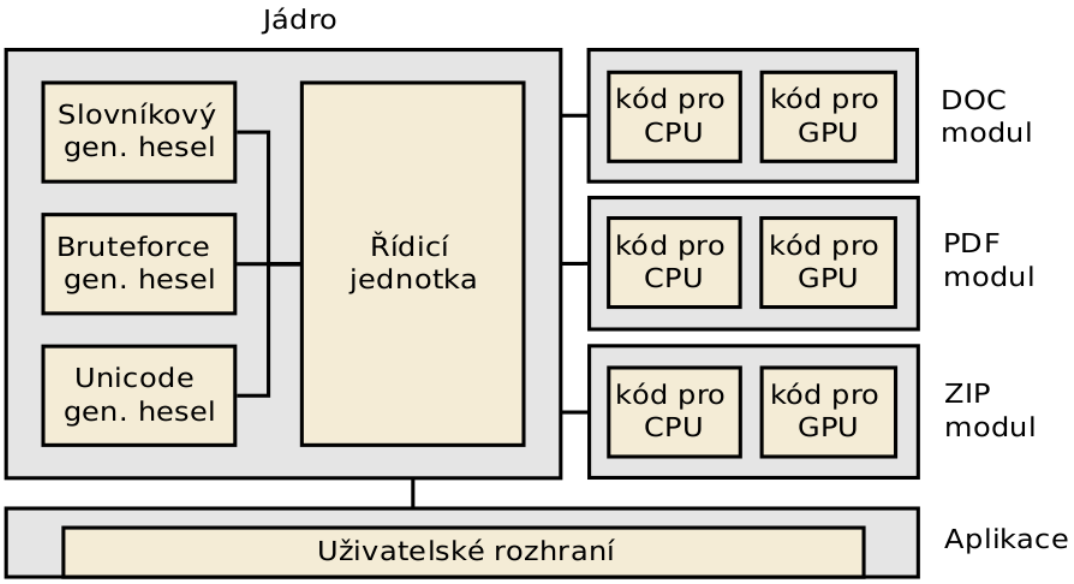
Hlavní cíle takového návrhu byly modularita, rozšiřitelnost a škálovatelnost. Schéma architektury nástroje Wrathion je vidět na obrázku 3.1.

3.1.1 Generátory hesel

Základním stavebním kamenem nástroje Wrathion je generátor hesel. Ten funguje na jednom ze tří principů:

- **Brute-force** – Generuje všechny možné kombinace povolených znaků.
- **Rule-based** – Tento generátor je ve formě návrhu. Funguje podobně jako brute-force, umožňuje však zanést do generování hesel jisté pravidla a generovat tak vhodnější hesla. Zohledňuje například nejčastěji se vyskytující digramy či trigramy v daném jazyku, čímž vznikají kombinace mnohem bližší skutečné řeči a zvyšuje se tedy pravděpodobnost na získání správného hesla.
- **Dictionary** – Hesla jsou generována z textového souboru, tzv. slovníku.

Po vygenerování hesla je toto předáno do třídy Cracker, která se stará o distribuci hesel mezi jednotlivé zařízení, na kterých probíhá ověřování, zda došlo ke shodě.



Obrázek 3.1: Architektura nástroje Wrathion

3.1.2 Zpracování na CPU

Pokud se rozhodneme nevyužívat OpenCL zařízení, ale celý proces provádíme přímo na CPU, jedná se o vcelku přímočarou sekvenci úkonů. Její schéma je na obrázku A.1.

3.1.3 Zpracování na GPU

Při použití GPU je nutné nejprve provést proces inicializace OpenCL zařízení, to znamená vytvoření kontextu, front příkazů, překlad kernelů a další.

Schéma prolamování hesel pomocí GPU je tedy mnohem složitější, je to však mnohem efektivnější varianta, která díky paralelnímu zpracování probíhá až o několik řádů rychleji. Schéma prolamování hesel pomocí GPU je možné vidět na obrázku A.2.

3.2 Podporované formáty

Wrathion v aktuální podobě podporuje následující formáty:

- **ZIP** – Modul pro tento formát podporuje šifrování pomocí AES a také dnes už zastaralý PKZIP.
- **PDF** – Modul pro PDF podporuje bezpečnostní revizi do verze 5.
- **DOC** – Jedná se o jediný formát ze skupiny dokumentů Microsoft Office. Podporuje pouze Microsoft Word verze 97-2000.

Jak již bylo zmíněno v kapitole 3.1, nástroj Wrathion byl od začátku navrhován k tomu, aby byl jednoduše rozšiřitelný. Kód pro každý formát je totiž v samostatném modulu. To nám umožňuje do frameworku přidat další moduly a rozšířit tak množinu formátů, které umí Wrathion prolomit. Každý modul obsahuje informaci potřebnou pro rozpoznání formátu a ověření, zda se opravdu jedná o daný formát.

Kapitola 4

OpenCL

Jedná se o standard, který umožňuje vývojářům plně využít potenciál paralelního zpracování informací na dnešních výpočetních jednotkách, jako jsou procesory nebo grafické karty. Výhodou je, že kód napsaný v tomto standardu je přenosný a použitelný na všech těchto platformách.

Jak bylo zmíněno výše, nástroj Wrathion tohoto standardu využívá k masivnímu urychlení celého procesu obnovy hesel. Je tedy nutné v tomto pokračovat a standard OpenCL použít i při návrhu nových modulů.

Použitý programovací jazyk se nazývá *OpenCL C* a je založen na standardu ISO/IEC 9899:1999, zkráceně C99. Je však rozšířen o množství funkcí, které jsou vhodné pro danou oblast [9]. Příkladem mohou být vektorové datové typy, datové typy pracující s vícerozměrnými poli, jako jsou obrázky, jejichž paralelním zpracováním se GPU zabývá velmi často, nebo kvalifikátory přístupových práv.

Architektura OpenCL je popsána následujícími modely: model platformy, exekuční model, paměťový model a programovací model [8].

4.1 Model platformy

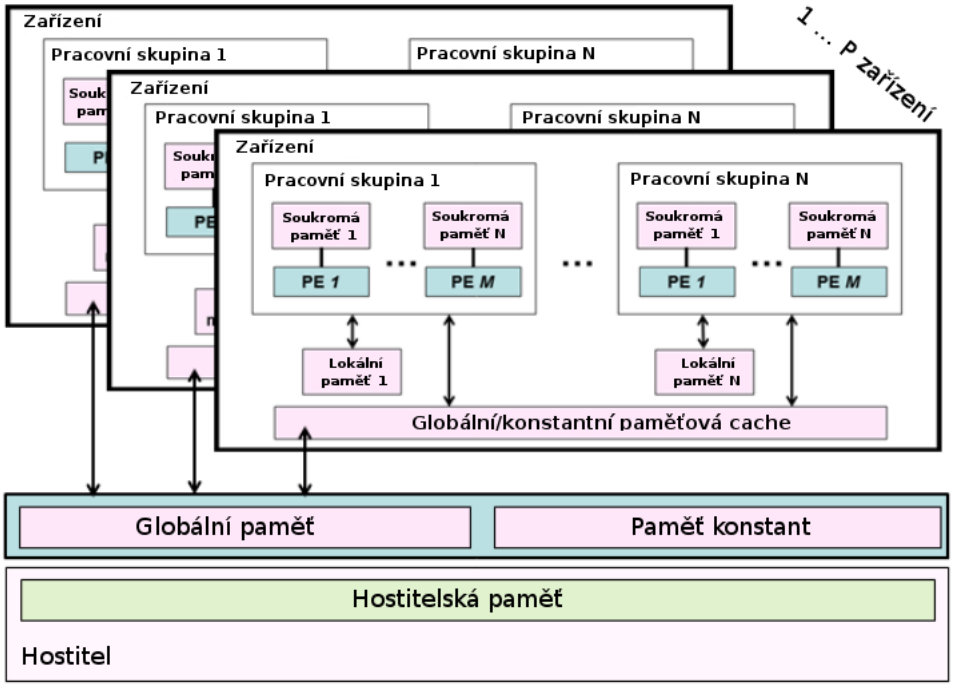
Pro práci s OpenCL potřebujeme hostitele, což je v našem případě klasický procesor. Ten musí být připojen k jednomu či více OpenCL zařízením, to znamená například GPU nebo více-jádrové CPU. To jsou zařízení schopná paralelního zpracování. Ta navíc obsahují množství výpočetních jednotek, kde každé dále obsahuje procesní prvky.

4.2 Exekuční model

Tento model se skládá z kernelu, což je konstrukce velmi podobná funkcím z jazyka C. Jedná se o základní jednotku, ve které je definováno, jak bude probíhat paralelní zpracování informací. Kolekce těchto kernelů je nazývána program.

Každá pracovní jednotka se nazývá *work-item*. Tyto jsou potom sdruženy do tzv. *work-groups*, tedy pracovních skupin, které jsou vykonávány hromadně na jednom zařízení a mohou mezi sebou sdílet lokální paměť a synchronizovat se.

Další částí tohoto modelu je aplikace, kterou představuje hostitel. Ta vytváří tzv. kontexty. V těch jsou zahrnuty informace o všech využívaných zařízeních, jejich parametrech a také kernelech a programech. Pro řízení se používají fronty požadavků, které zasílají do zařízení s možností paralelního zpracování příkazy. Tyto fronty mohou být zpracovávány



Obrázek 4.1: Schéma paměťového modelu [8]

buď *in-order*, což znamená, že příkazy ve frontě čekají, až se dokončí předchozí příkaz, nebo *out-of-order*, což umožňuje provedení čekajících příkazů v jakémkoliv pořadí.

4.3 Paměťový model

Tento model se zabývá různými typy paměti, se kterou je možné při práci s OpenCL operovat. Můžeme zde identifikovat dvě základní skupiny. Hostitelská paměť, se kterou lze pracovat dle očekávání v rámci hostitele a paměť zařízení, se kterou pracují kernely. Tu lze dále rozdělit na 4 části.

- **Globální paměť** – Do této části mají přístup všechny kernely v rámci všech zařízení v daném kontextu.
- **Paměť konstant** – Jedná se o část globální paměti, která se v průběhu vykonávání programu nemění.
- **Lokální paměť** – Je to region paměti, ke kterému mají přístup všechny pracovní položky v lokální pracovní skupině.
- **Soukromá paměť** – Toto je paměť, se kterou pracuje pouze daná pracovní položka.

Je nutné brát ohled na to, že paměť zařízení nelze alokovat dynamicky. Všechno potřebné místo je tedy vhodné alokovat dopředu. Komunikace mezi hostitelem a zařízením pak probíhá klasicky pomocí fronty.

Cělý paměťový model názorně zobrazuje obrázek 4.1.

Kapitola 5

Struktura dokumentů MS Office

V této kapitole bude podrobněji rozebrána struktura jednotlivých formátů MS Office, jejíž znalost je nutná pro získání informací, potřebných pro obnovu hesla.

Jediný formát, který Wrathion v současné době ze skupiny dokumentů MS Office podporuje, je DOC. Ten je určen pro nástroj Microsoft Word a je podporována pouze verze 97-2000. Od té doby však vyšlo množství nových verzí, a výše zmíněná verze se již téměř nepoužívá. Složitost šifrování je navíc v každé další verzi na vyšší úrovni.

Dalším formátem vhodným pro dodání k nástroji Wrathion je formát nástroje Microsoft Excel, tedy XLS, a Microsoft PowerPoint, PPT.

5.1 Balík Microsoft Office

Jedná se o skupinu programů vyvinutou společností Microsoft, která je určená pro různou kancelářskou práci. Celý balík zahrnuje více než 10 různých nástrojů, my se však v rámci tohoto projektu zaměříme pouze na tři nejrozšířenější nástroje, protože pouze tyto podporují možnost zašifrování.

- **Microsoft Word** – Jedná se o nástroj pro editaci textů, který je velmi rozšířený. Jeho formáty jsou DOC a DOCX.
- **Microsoft Excel** – Je to nástroj pro vytváření a editaci tabulek. Jeho formáty jsou XLS a XLSX.
- **Microsoft PowerPoint** – Jedná se o nástroj pro vytváření prezentací. Jeho formáty jsou PPT a PPTX.

Balík programů Microsoft Office v posledních letech přišel také na operační systémy Mac a Android. V tomto projektu bereme však v úvahu pouze platformu Windows, která patří mezi nejrozšířenější.

Balíky Microsoft Office vycházejí v nových verzích každých několik let. Důležité je, že společnost Microsoft stále vylepšuje zabezpečení těchto formátů, a proto se často s novou verzí mění také metoda zabezpečení [4].

- **Verze 95 a starší** – Tyto verze používají pro zabezpečení pouze 16-bitový klíč. Obnova hesel je u těchto formátů tedy okamžitá a v rámci tohoto projektu se jimi nebudeme zabývat.

- **Verze 97-2000, XP a 2003** – Z těchto verzí je momentálně nástrojem Wrathion podporována pouze formát DOC verze 97-2000. V rámci tohoto projektu se zaměříme na vytvoření modulů pro formáty XLS a PPT a úpravu DOC modulu pro podporu všech těchto verzí.
- **Verze 2007, 2010, 2013, 2016** – Tyto verze se vyznačují novou strukturou, tzv. Office Open XML (viz kapitolu 5.6), jedná se tedy o formáty DOCX, XLSX a PPTX. Navíc bylo vylepšeno zabezpečení a místo hešovací funkce MD5 a šifrovacího algoritmu RC4 byly použity funkce SHA-1, respektive SHA-2, a AES. Proces obnovení hesla se tedy oproti předchozím verzím rapidně zpomalí.

5.2 Šifrování MS Office

Balík Microsoft Office podporuje několik typů zašifrování dokumentu.

- **XOR Obfuskace** – Tato metoda provede transformaci streamů dat na místě pomocí operace exkluzivní disjunkce. Je možné ji použít u formátů DOC a XLS. V praxi se však nedoporučuje využívat, protože je jednoduše prolomitelná. V nových verzích se však tato metoda nabízí kvůli zpětné kompatibilitě verzí.
- **40-bitová šifra RC4** – Jak napovídá název, je použita RC4 s délkou klíče 40 bitů. Všechny podstatné informace jsou uloženy ve struktuře *EncryptionHeader*. Je využita ve verzi Office 97-2000.
- **Šifra RC4 s CryptoAPI** – V této metodě je opět standardně použita RC4. *EncryptionHeader* má ovšem pozměněnou strukturu. Tato metoda byla zavedena s verzí XP a 2003 a předchozími verzemi (97-2000) není podporována.
- **ECMA-376 šifrování** – Tato metoda umožňuje použít různé šifrovací a hešovací algoritmy, včetně použití externích „poskytovatelů“ zabezpečení. Informace o šifrování jsou uloženy ve speciálním streamu *EncryptionInfo*, který může být jak v binárním, tak XML formátu. Je použita u formátů Office Open XML (2007 a novější).

V této práci budeme brát v úvahu všechny metody kromě XOR obfuskace, která je, jak již bylo řečeno, nepoužívaná a jednoduše prolomitelná.

5.3 Struktura formátu DOC

Jedná se o formát, užívaný nástrojem Microsoft Word a to do verze 2003. Jedná se o tzv. Office Binary File Format, a je inspirován souborovým systémem FAT (tzv. OLE formát [2]). Dokument tohoto formátu je rozdělen na jednotlivé sektory [3]. Prvním sektorem je hlavička, ve které je stanoveno, jaká je velikost jednoho sektoru, v jakém sektoru začíná datový tok, apod. Těchto informací využijeme i ve formátech XLS a PPT.

Word Binary File Format je možné zašifrovat 3 možnými způsoby:

- **XOR Obfuskace**,
- **40-bitová šifra RC4**,
- **šifra RC4 s CryptoAPI**.

Ve verzi 97-2000 je užita 40-bitová šifra RC4 a hešovací funkce MD5, ve verzi XP je heslo zahešováno opět funkcí MD5, ale je možné využít metodu CryptoAPI a ve verzi 2003 je pak použita jako hešovací funkce SHA-1.

Struktura formátu DOC je následující u všech verzí. Na začátku hlavního datového toku (*WordDocument Stream*) se nachází tzv. FIB (*File Information Table*). V této tabulce jsou uloženy všechny informace o souboru. Nás bude zajímat prvních 32 bytů této struktury, tzv. *FibBase*. V tomto prostoru je uložena informace o tom, zda je soubor zašifrován a jakým způsobem. Také zde nalezneme odkaz do datové oblasti kde se nachází tzv. *EncryptionHeader*, který obsahuje námi hledané hodnoty. Ty se zde nachází ve formě následujících 16-bitových hodnot.

- **Salt** – Jedná se o náhodně vygenerovanou hodnotu, která se spolu s heslem použije pro vytvoření klíče.
- **EncryptedVerifier** – Je to posloupnost dat, užívaná pro ověření správnosti hesla, následně zašifrovaná zvoleným algoritmem. Jejím generováním se budeme zabývat níže.
- **EncryptedVerifierHash** – Jedná se o zašifrovaný heš ověřující posloupnosti dat (viz předchozí odrážku).

5.4 Struktura formátu XLS

Jedná se o formát, užívaný nástrojem Microsoft Excel, od verze 97-2000 až do verze 2003. Zajímavostí je, že maximální délka hesla u těchto verzí je 15 znaků.

Excel Binary File Format lze zašifrovat opět 3 způsoby, stejně jako u předchozího formátu:

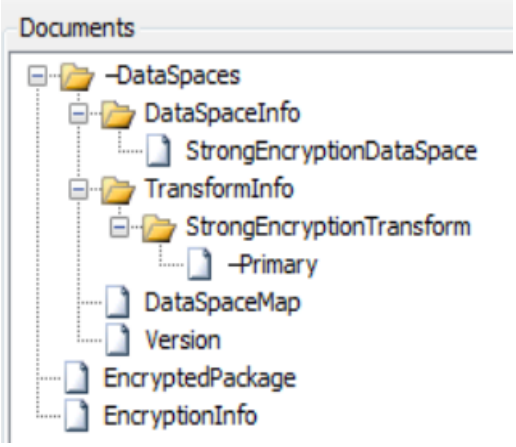
- **XOR Obfuskace**,
- **40-bitová šifra RC4**,
- **šifra RC4 s CryptoAPI**.

Kromě OLE hlavičky, která je popsána výše, se tento formát skládá z množství tzv. *streamů*, *substreamů* a *záznamů* [6]. Hlavním streamem v dokumentech XLS je *Workbook*, který obsahuje množinu substreamů pro každý spreadsheet. Každý substream začíná speciálním záznamem BOF (*Beginning Of File*) a končí speciálním záznamem EOF (*End Of File*). Prvním substreamem je však pokaždé substream *Globals*. Ten obsahuje informace o datech a vlastnostech dokumentu. Tento bude pro nás důležitý. Stejně jako všechny substreamy, také substream *Globals* obsahuje množinu záznamů. Obecně každý záznam začíná 4 byty, informujícími o typu záznamu (Record Type – rt) a jeho délce (Count of Bytes – cb). Důležitým záznamem je pro nás *FilePass* (rt = `0x002F`), který se nachází na druhém místě tohoto substreamu, hned po záznamu *WriteProtect*. Pokud záznam *FilePass* v substreamu *Globals* existuje, je dokument zašifrovaný.

5.5 Struktura formátu PPT

Jedná se o formát, užívaný nástrojem Microsoft PowerPoint, opět až do verze 2003.

Dokumenty s formátem PPT podporují pouze jedinou metodu šifrování:



Obrázek 5.1: Náhled struktury zaheslovaného dokumentu

- **Šifra RC4 s CryptoAPI**

Struktura formátu PPT je opět tvořena souborovým systémem OLE. Hlavními streamy v těchto dokumentech jsou *Current User Stream*, ve kterém jsou uloženy informace o aktuálním, respektive posledním uživateli, a *PowerPoint Document Stream*, ve kterém jsou informace o dokumentu [5]. Tyto streamy obsahují množství záznamů, které se jmenují *atomy*, respektive *kontejnery*. Na začátku všech takových záznamů se nachází tzv. *RecordHeader*, který má typicky 8 bytů a informuje o typu záznamu (*rt*), jeho délce (*rl*), apod. Důležitý pro nás bude *CryptSession10Container*, který obsahuje informace o šifrování ve formě struktury *EncryptionHeader*. Z té je možné získat všechny potřebné informace k obnově hesla.

5.6 Office Open XML formát

Jedná se o specifikaci souborového formátu vyvinutého společností Microsoft. Poprvé byl použit v balíku Office 2007. Office Open XML formát byl standardizován pod záštitou společnosti Ecma, jejíž členové jsou firmy jako Intel, Apple, Microsoft, apod. Standard je pětidílný a velice obsáhlý [1]. Vznikl kvůli potřebě sjednotit a zjednodušit formát předchozích verzí dokumentů Office. Jak již bylo zmíněno výše, soubory Office v tomto formátu mají dodatečnou příponu „X“, tedy DOCX, XLSX a PPTX.

Tyto formáty jsou složeny z množství streamů, v binárním nebo XML formátu, které mají určitou strukturu, lišící se podle verze balíku Office. Tyto soubory jsou následně zabaleny nástrojem ZIP a jednotlivé soubory potom zašifrovány [20]. To znamená, že není šifrován celý archiv. Struktura šifrovaného a nešifrovaného archivu je zcela odlišná. My se zaměříme na strukturu šifrovaného archivu.

Důležité je, že pro všechny tři zpracovávané formáty platí, že struktura šifrovaného souboru se neliší. Můžeme tedy zpracovávat všechny stejně. V kořenovém adresáři po rozbalení se nachází několik složek s informacemi o šifrovaném souboru, dále pak stream *EncryptedPackage*, obsahující samotná zašifrovaná data, a soubor *EncryptedInfo*. Ten je pro nás důležitý, protože obsahuje informace o použité hešovací funkci, šifrovacím algoritmu, použitém saltu a dalších hodnotách potřebných pro obnovu hesla.

Náhled struktury archivu je vidět na obrázku 5.1.

Tabulka 5.1: Přehled standardního zabezpečení dokumentů Office

Verze Office	Hešovací funkce	Šifrovací algoritmus	Délka klíče
97-2000, XP	MD5	RC4	40b
2003	SHA-1	RC4	40-128b
2007	SHA-1 (50 000krát)	AES	128b
2010	SHA-1 (100 000krát)	AES	128b
2013, 2016	SHA-2 (100 000krát)	AES	256b

5.7 Porovnání formátů

Pokud bychom porovnali výše popsané formáty, zjistili bychom, že dokumenty Office do verze 2003 včetně nejsou pro počítače dnešního výkonu problém a jednoduchá uživatelská hesla dokáží získat během několika sekund až minut. Použité algoritmy jako jsou RC4 nebo MD5 jsou již dnes považovány za překonané a není doporučené je používat. Od roku 2007 se však úroveň zabezpečení rapidně zvýšila díky zavedení šifrovacího algoritmu AES a použití iterativního hešování. Každou novou verzí je navíc toto zabezpečení zvýšeno použitím novějšího standardu. Pro dnešní výpočetní výkon je zabezpečení od verze 2013 pro brute-force útoky téměř neprolomitelné.

Co se týče struktury dokumentu, do verze Office 2003 včetně byla struktura všech tří zmiňovaných formátů odlišná a orientace v těchto formátech složitá. Zavedením Open XML formátu byly formáty sjednoceny a získávání informací z XML souborů je přímočaré.

Porovnání použitých algoritmů v jednotlivých verzích je možné vidět v tabulce 5.1.

Kapitola 6

Návrh nových modulů

Cílem této práce je navrhnout a implementovat moduly pro nové formáty dokumentů Microsoft Office do nástroje Wrathion. Jak již bylo zmíněno, nástroj Wrathion je od začátku navrhován jako modulární a rozšiřitelný. Moduly pro jednotlivé části programu jsou tedy relativně samostatné a jsou načítány z dynamických knihoven.

Nástroj Wrathion již obsahuje několik modulů a při návrhu nových modulů bude postupováno podle této předlohy. O vnitřní záležitosti jako je načítání souboru, generování hesel, či výběr správného modulu, se stará jádro aplikace. Na nás je navrhnout, co provést s vygenerovaným heslem a s čím tuto hodnotu porovnat. Při získávání informací ze souboru je nutné postupovat podle struktury popsané v kapitole 5 a při transformaci generovaného hesla využijeme znalosti hešovacích funkcí a šifrovacích algoritmů popsaných v kapitole 2.

Pokud chceme vytvořit či upravit nový modul, musíme znát jeho příponu, například *.doc*, *.docx*, apod., a signaturu souboru. To je v našem případě maximálně 32 bytů dlouhá posloupnost, vyskytující se na začátku souboru. Po analýze těchto hodnot je soubor rozpoznán a může být zavolán příslušný modul.

Dále je nutné vytvořit tzv. *cracker*, který se bude starat o samotný proces transformace generovaného hesla do podoby vhodné pro ověření. Tento cracker může běžet buď na CPU, nebo na GPU, kde bude využito standardu OpenCL (viz kapitolu 4). Rozdíl mezi těmito variantami je rozebrán v sekci 3.1.

6.1 Modul DOC

Tento modul již v nástroji Wrathion existuje, podporuje však pouze verzi 97-2000. Přípona *.doc* a signatura `0xD0CF11E0A1B11AE1` je však stejná i u dalších verzí dokumentů Office, kde je tento formát použit, tedy XP a 2003. Také implementované algoritmy RC4 a MD5 se u většiny verzí neliší. Je tedy vhodné tento modul použít a pouze jej rozšířit.

Při získávání informací z DOC souborů začneme u hlavičky OLE formátu, která se vyskytuje vždy na počátku souboru. Nalezneme zde například signaturu souboru, či informace o verzi. Důležitou hodnotou je pro nás tzv. *SectorShift*, který má velikost 2 byty a leží na offsetu `0x001E`. Tato hodnota určuje velikost jednoho sektoru. Dále také zjistíme index prvního directory sektoru, ve kterém budeme hledat *EncryptionHeader* (viz výše). Ten je uložen jako 4-bytová hodnota *Directory Starting Sector Location* na offsetu `0x0030`.

Po přečtení OLE hlavičky můžeme přejít na hlavní datový stream – *WordDocument stream*, ležící typicky v prvním sektoru. Prvních 32 bytů tohoto sektoru je *FibBase*. Pro tuto konstrukci byla vytvořena stejnojmenná struktura, do které se celý *FibBase* načte a

z ní zjistíme jen potřebné informace. Mezi ně patří bit *fEncrypted*, jehož nastavení určuje, zda je soubor šifrován, a bit *fObfuscation*. Pokud je tento nastaven na 1, je použita XOR obfuskace, pokud je nulový, je využito jedné ze dvou zbývajících metod popsaných výše. Dále je nutné přečíst bit *fWhichTblStm*, který určuje, zda *EncryptionHeader* leží ve streamu *0Table* nebo *1Table*.

Po nalezení jednoho z těchto streamů se dostaneme ke struktuře *EncryptionHeader*, která leží vždy na nulovém offsetu. Na počátku této struktury se nachází dvojice 16-bitových hodnot, tzv. *EncryptionVersionInfo*. Pokud jsou obě tyto hodnoty rovny `0x0001`, jedná se o metodu 40-bitového RC4. Ta už je v nástroji Wrathion implementována a nebudeme se jí zde více zabývat.

6.1.1 CryptoAPI EncryptionHeader

V opačném případě byla použita metoda s CryptoAPI. První hodnota této struktury – *vMajor* musí být rovna `0x0002` (v případě verze 2003), `0x0003` (v případě uložení ve verzi 2007 SP1) nebo `0x0004` (v případě 2007 SP2 a novějších verzí) a druhá hodnota *vMinor* musí být rovna `0x0002`. Následuje struktura *EncryptionHeader*, v níž jsou uloženy informace o použitém algoritmu. V tomto případě však musí být šifrovací algoritmus nastaven na RC4 a hešovací funkce na SHA-1. Poté následuje struktura *EncryptionVerifier*. Ta obsahuje 32-bitovou hodnotu *SaltSize*, která musí být rovna `0x00000010`. Dále je zde 16-bytová hodnota *Salt* následovaná 16-bytovým *EncryptedVerifier* a 4-bytovou hodnotou *VerifierHashSize*, určující kolik bytů se použije z následující 20-bytové hodnoty *EncryptedVerifierHash* po jejím odšifrování.

Po získání těchto hodnot můžeme přejít na transformaci vygenerovaného hesla do podoby vhodné pro ověření. Nejprve se provede konkatenace hesla v kódování Unicode s 16-bytovou hodnotou *Salt* a tento řetězec se zahešuje funkcí SHA-1. Poté se provede konkatenace tohoto heše s 32-bitovou hodnotou poskytnutou aplikací (v našem případě to bude blok `0x00000000`) a tento řetězec je opět zahešován. Prvních *KeySize* bitů (hodnota daná ve struktuře *EncryptionHeader*) je považováno za šifrovací klíč. Ten se poté použije k dešifrování hodnoty *EncryptedVerifier*, čímž je získán 16-bytový *Verifier* a hodoty *EncryptedVerifierHash*, čímž je získán 20-bytový heš předchozí hodnoty. Poté stačí zahešovat získanou hodnotu *Verifier* a porovnat ji s dešifrovaným hešem. Pokud se hodnoty nerovnají, heslo je nesprávné. V opačném případě byla obnova hesla úspěšná.

6.2 Modul XLS

Tento modul řeší obnovu hesel dokumentů, které mají příponu *.xls* a signaturu stejnou jako DOC, tedy `0xD0CF11E0A1B11AE1`.

Existuje opět několik možností zašifrování souboru XLS, viz sekci 5.4. To, která je použita, je dáno v prvních 16 bitech záznamu *FilePass*, v položce *wEncryptionType*.

- **XOR obfuskace** – Je dána hodnotou 0x0000.
- **Šifra RC4/RC4 s CryptoAPI** – Je dána hodnotou 0x0001.

Záznam *FilePass* začneme hledat v datovém sektoru substreamu *Globals*, dokud nenarazíme na hodnotu `0x2F00`, značící začátek záznamu. Pokud narazíme na hodnotu `0x0A`, značící konec substreamu *Globals*, záznam se v dokumentu nenachází a soubor tedy není šifrován. Kromě toho můžeme hledání zkrátit podle délky offsetu od začátku substreamu. Záznam

FilePass musí totiž ležet v několika prvních bytech substreamu, v závislosti na přítomnosti několika volitelných záznamů. Pokud jsme tedy prohledali 100 bytů a záznam jsme nenašli, můžeme soubor prohlásit za nešifrovaný.

Po nalezení záznamu *FilePass* přečeteme jeho velikost uloženou jako 2-bytová hodnota a poté ověříme, zda je v prvních 16 bitech uložená hodnota `0x0001`, která značí užití metody RC4. Na následující pozici se nachází struktura *EncryptionHeader*, buď ve verzi CryptoAPI, z níž získáme potřebné informace identicky, jak bylo popsané v kapitole 6.1.1, nebo ve verzi klasické 40-bitové šifry RC4.

6.2.1 RC4 EncryptionHeader

Pokud jsou tedy následující dvě 16-bitové hodnoty rovny `0x0001` a `0x0001`, jedná se o metodu šifrování RC4. V tomto případě použijeme algoritmus, který je již implementován pro současný modul DOC verze 97-2000. V hlavičce *EncryptionHeaderu* se po informacích o verzi nachází po sobě jdoucí tři 16-bitové hodnoty – *Salt*, *EncryptedVerifier* a *EncryptedVerifierHash*.

Po přečtení těchto hodnot můžeme opět přistoupit k transformaci vygenerovaného hesla. Tato transformace se liší od postupu výše. První krok je zahešování hesla v kódování Unicode hešovací funkcí MD5. Prvních 5 bytů získaného heše se konkatenuje s 16-bytovou hodnotou *Salt*. Tato 21-bytová hodnota se zkopíruje 16x za sebe do 336-bytového bufferu. Jeho obsah je poté opět zahešován funkcí MD5. Z tohoto heše je opět použito prvních 5 bytů, které jsou zkonkatenovány s 32-bitovým blokem dat (v našem případě to bude blok `0x00000000`) a výsledek se naposledy zahešuje. Prvních 128 bitů tohoto heše je považováno za šifrovací klíč.

Ověření, zda je heslo správné je pak stejné jako v předchozím případě. Pomocí šifrovacího klíče dešifrujeme hodnotu *EncryptedVerifier* a *EncryptedVerifierHash*. Důležité je, že RC4 stream nesmí být mezi těmito dvěma akcemi resetován. Poté zahešujeme získaný *Verifier* a porovnáme získaný výsledek se získaným *VerifierHash*. Pokud se výsledky shodují, obnova hesla byla úspěšná.

6.3 Modul PPT

Tento modul řeší obnovu hesel dokumentů s příponou *.ppt*. Signatura se opět shoduje s předchozími formáty – `0xD0CF11E0A1B11AE1`.

Jak již bylo zmíněno v sekci 5.5, v celém dokumentu nás bude zajímat pouze kontejner *CryptSession10Container*. Jeho *rt* se rovná hodnotě `0x2F14` a můžeme jej v souboru identifikovat posloupností následujících bytů celého *RecordHeaderu* – `0x0F00142F`. Poté následují 4 byty, informující o délce kontejneru, a dále již samotná data, která obsahují *CryptoAPI EncryptionHeader*. Z toho je možné získat potřebné hodnoty a provést ověření hesla podle postupu popsaného v podsekci 6.1.1.

6.4 Modul OOXML

Tento modul bude zahrnovat všechny Office Open XML formáty, tedy *.docx*, *.xlsx* a *.pptx*. To je možné díky tomu, že jako zašifrované soubory mají všechny tři formáty stejnou strukturu. Signatura všech těchto formátů je `0x504B030414000600`. To však platí pouze pro nezašifrované dokumenty, které nás v rámci této práce nezajímají. Zašifrované dokumenty formátu OOXML mají stejnou signaturu jako všechny předchozí formáty, tedy

`0xD0CF11E0A1B11AE1`. Získávání potřebných informací z Open XML dokumentů bude mnohem jednodušší než u předchozích binárních formátů. Všechny potřebné informace, tedy použitý hešovací a šifrovací algoritmus, salt i ověřovací hodnoty, nalezneme ve streamu *EncryptionInfo*, buď v binární formě nebo jako XML.

Různé verze dokumentů Office mají různé standardní šifrovací a hešovací algoritmy a navíc povolují užití množství dalších algoritmů. Proto bude nutné vytvořit několik různých crackerů, pro všechny tyto algoritmy.

Jak již bylo popsáno výše, Office Open XML dokumenty jsou šifrovány pomocí metody ECMA-376. Tu je možné dále rozdělit do 3 skupin.

- **Standardní šifrování** – Informace o šifrování jsou uloženy v binární podobě ve streamu *EncryptionInfo*. Dovoluje použití šifrovacího algoritmu AES (128b, 192b nebo 256b) a hešovací funkce SHA-1. Metoda je použita ve verzi Office 2007.
- **Agilní šifrování** – Informace o šifrování jsou uloženy ve formě XML ve streamu *EncryptionInfo*. Narozdíl od předchozí metody dovoluje použití široké škály šifrovacích i hešovacích algoritmů. Metoda je použita od verze Office 2010 a dále.
- **Rozšiřitelné šifrování** – Tato metoda dovoluje použití externích poskytovatelů šifrování a v této práci ji nebudeme zohledňovat.

6.4.1 Standardní šifrování

Při použití této metody nalezneme všechny informace o šifrování ve streamu *EncryptionInfo* a to v binární formě, ve struktuře *EncryptionHeader*, která se velmi podobá struktuře popsané v podsekci 6.1.1. Použití této metody je možné rozpoznat podle verze této struktury – 16-bitová hodnota *vMajor* musí být rovna `0x0002`, `0x0003` nebo `0x0004` a hodnota *vMinor* musí být rovna hodnotě `0x0002`. Následuje 32-bitová struktura *EncryptionHeaderFlags*, v níž musí být nastaven bit *fCryptoAPI* a *fAES*. Po přečtení 32-bitové hodnoty udávající velikost následující struktury se dostaneme na *EncryptionHeader*. V té je uložen, kromě jiného, algoritmus použitý pro šifrování a hešování. 32-bitová položka *AlgID*, ležící na offsetu `0x0040`, musí nabývat hodnot `0x0000660E` pro AES-128, `0x000000C0` pro AES-192 nebo `0x00006610` pro AES-256. V následující hodnotě, která udává hešovací algoritmus musí být uložena hodnota `0x00008004`, značící SHA-1. Velikost klíče *KeySize* je dána číslem za zkratkou AES. Po zjištění techto hodnot můžeme zbytek informací ve struktuře *EncryptionHeader* přeskočit.

Následuje struktura *EnrcyptionVerifier*, ze které získáme hodnoty potřebné pro ověření hesla, tedy *Salt*, *EncryptedVerifier* a *EncryptedVerifierHash*, a to stejným způsobem, jako je popsáno v podsekci 6.1.1.

$$H_n = (iterator + H_{n-1}) \tag{6.1}$$

Proces generování šifrovacího klíče je následující. Nejprve se provede konkatenace 16-bytové hodnoty *Salt* s vygenerovaným heslem v kódování Unicode a výsledný řetězec je zahešován funkcí SHA-1. Následně je provedeno 50 000 iterací této hešovací funkce nad řetězcem, který vznikne konkatenací tzv. *iterátoru*, což je lineárně se zvyšující 32-bitová hodnota, počínající na `0x00000000`, a výsledkem předchozího hešování, viz rovnici 6.1, kde H je hešovací funkce SHA-1. To znamená, že pro poslední cyklus bude *iterátor* nabývat hodnoty 49 999. Poté se provede konkatenace výsledného heše s 32-bitovým blokem dat

(v našem případě to bude blok `0x00000000`) a výsledek je naposled zahešován funkcí SHA-1. Na rozdíl od předchozích algoritmů, toto však ještě není šifrovací klíč a s výsledným hešem se musí dále pracovat.

1. Vytvoříme 64-bytový buffer, do kterého zkopírujeme 64x byte o hodnotě `0x36`. Nad prvními 20 byty bufferu provedeme operaci exklusivního součtu s výsledným hešem a tento řetězec zahešujeme. Získáme tak 20-bytový heš, který nazveme **X1**.

2. Opět vytvoříme 64-bytový buffer, do něhož zkopírujeme 64x byte o hodnotě `0x5C`. Opět provedeme operaci XOR s prvními 20 byty bufferu a výsledným hešem a tento řetězec zahešujeme. Vznikne tak 20-bytový heš, který nazveme **X2**.

3. Provedeme konkatenaci předchozích hodnot a získáme tak 40-bytovou hodnotu **X3**.

$$X3 = X1 + X2 \tag{6.2}$$

4. Prvních *KeySize* bitů hodnoty X3 je považováno za šifrovací klíč.

Následné ověření správnosti hesla se prakticky neliší od předchozích verzí. Nejprve dešifrujeme hodnoty *EncryptedVerifier* a *EncryptedVerifierHash*. Důležité je, že musí být použit režim ECB, viz podsekci 2.2.4. Poté je dešifrovaná hodnota *Verifier* zahešována funcí SHA-1 a výsledek porovnán s dešifrovanou hodnotou *VerifierHash*. Pokud se hodnoty rovnají, obnova hesla byla úspešná.

6.4.2 Agilní šifrování

Tato metoda byla poprvé použita ve verzi Office 2010. Informace o šifrování jsou opět uloženy ve streamu *EncryptionInfo*, ovšem nikoli v binární formě, jako tomu bylo doposud, ale v XML. Před samotným XML se opět nachází dvě 16-bitové hodnoty informující o verzi. Pro metodu agilního šifrování je hodnota *vMajor* i *vMinor* rovna `0x0004`. Po rezervovaném 4-bytovém místě následuje samotné XML. To se skládá z množství záznamů. Nás bude pro potřebu obnovy hesla zajímat složený atribut *keyEncryptors*, který obsahuje údaje o vytvoření šifrovacího klíče. Pro ověření hesla budeme potřebovat následující atributy.

- **saltSize** – Specifikuje velikost hodnoty *Salt* v bytech.
- **blockSize** – Značí velikost jednoho šifrovaného bloku.
- **keyBits** – Značí délku šifrovacího klíče v bitech.
- **hashSize** – Jedná se o délku výstupního heše, která odpovídá použité hešovací funkci.
- **spinCount** – Udává počet iterací hešovací funkce. Standardně jde o hodnotu 100 000, může však nabývat hodnoty až 10 000 000.
- **cipherAlgorithm** – Specifikuje použitý šifrovací algoritmus. V metodě agilního šifrování lze použít následující algoritmy:
  - **AES (128/192/256)**,
  - **RC2**,
  - **DES**, **DESX**, **3DES**, **3DES_112**.

- **cipherChaining** – Specifikuje, jaký režim používá šifrovací algoritmus. Je možné použít následující režimy:
  - **ChainingModeCBC** – Cipher Block Chaining (CBC),
  - **ChainingModeCFB** – Cipher Feedback Chaining (CFB).
- **hashAlgorithm** – Udává použitý hešovací algoritmus. Je možné použít jeden z následujících algoritmů:
  - **SHA-1**, **SHA256**, **SHA384**, **SHA512**,
  - **MD2**, **MD4**, **MD5**,
  - **RIPEMD-128**, **RIPEMD-160**,
  - **WHIRLPOOL**.
- **saltValue** – Specifikuje náhodně vygenerovanou hodnotu *Salt*, v kódování base64[1].
- **encryptedVerifierHashInput** – Jedná se vlastně o položku *EncryptedVerifier*, známou z výše popsaných metod. V dokumentu je uvedena v kódování base64.
- **encryptedVerifierHashValue** – Jedná se o hodnotu *EncryptedVerifierHash*, známou z výše popsaných metod. Opět je zde v kódování base64.

Proces generování šifrovacího klíče je obdobný jako u metody standardního šifrování. Rozdíl je však v tom, že k zašifrování hodnoty *encryptedVerifierHashInput* se použije jiný klíč, než k zašifrování hodnoty *encryptedVerifierHashValue*. Způsob vytvoření šifrovacích klíčů je následující.

Nejprve je provedena konkatenace hodnoty *Salt* a vygenerovaného hesla v kódování Unicode. Před tím je nutné převést hodnotu *Salt* z kódování base64 do binární podoby. Výsledek je zahešován hešovací funkcí danou hodnotou *hashAlgorithm*. Poté je provedeno iterativní zahešování tohoto řetězce podle způsobu popsaného rovnicí 6.1. Maximální hodnota iterátoru v posledním cyklu hešování bude dána hodnotou *spinCount – 1*. Poté se provede konkatenace získaného heše s hodnotou *blockKey*. Ta je pevně daná rozdílnými hodnotami pro oba klíče.

- `0xFE, 0xA7, 0xD2, 0x76, 0x3B, 0x4B, 0x9E` a `0x79` – Toto je posloupnost bytů hodnoty *blockKey* pro generování šifrovacího klíče pro *encryptedVerifierHashInput*.
- `0xD7, 0xAA, 0x0F, 0x6D, 0x30, 0x61, 0x34` a `0x4E` – Toto je posloupnost bytů hodnoty *blockKey* pro generování šifrovacího klíče pro *encryptedVerifierHashValue*.

Výsledek konkatenace je naposled zahešován. Pokud je délka výsledného řetězce menší než potřebná délka klíče, dána hodnotou *keyBits*, je tento řetězec doplněn na požadovanou délku přidáním bytů s hodnotou `0x36`. Pokud je délka heše větší, jednoduše použijeme pouze prvních *keyBits* bitů.

Nyní již stačí dekódovat hodnoty *encryptedVerifierHashInput* a *encryptedVerifierHashValue* do binární podoby a dešifrovat je pomocí algoritmu specifikovaném v *cipherAlgorithm* v režimu *cipherChaining*. Jako IV bude použita binární podoba *saltValue*, doplněna byty `0x00` tak, aby se její velikost rovnala *blockSize*. Poté stačí dešifrovanou hodnotu *VerifierHashInput* zahešovat a výsledek porovnat s dešifrovanou hodnotou *VerifierHashValue*. Pokud se hodnoty rovnají, obnova hesla byla úspěšná.

[1] Převádí 3 binární oktety na 4 tisknutelné znaky, podle speciální tabulky.

Kapitola 7

Implementace nových modulů

V této kapitole budou rozebrány detaily samotné implementace jednotlivých modulů, nových i upravených, ale také způsob, jakým jsou nové moduly začleněny do již existujícího nástroje.

V původní podobě obsahoval nástroj Wrathion pouze tři moduly, které spolu souvisely jen vzdáleně a nebyla tedy nutná spolupráce těchto modulů. V případě tvorby modulů pro dokumenty Office mají ale tyto formáty jisté souvislosti a určité funkce, které budou popsány níže, jsou dokonce identické. Před vytvářením nových modulů jsme tedy museli vzít v úvahu tento fakt, abychom se vyhnuli zbytečné redundanci kódu. To by vyústilo ve špatnou čitelnost zdrojových kódů a špatnou udržovatelnost celého nástroje.

Z tohoto důvodu byla vytvořena složka *OfficeSources*, která obsahuje všechny zdrojové kódy, hlavičkové soubory a kernely, sdílené mezi více moduly. Z těch je následně vytvořena dynamická knihovna, která se při kompilaci přilinkuje k jednotlivým modulům.

7.1 Úprava modulu DOC

Při implementaci bylo postupováno dle návrhu. To znamená, že jsme využili již existujícího modulu a provedli jsme jeho rozšíření na verze XP a 2003. Nejprve však byla provedena restrukturalizace.

Vzhledem k tomu, že formáty DOC, XLS a PPT se liší pouze ve způsobu uložení šifrovacích informací, bylo možné pro všechny tyto tři formáty použít stejný cracker. Proto byly již existující soubory *DOCRC4CrackerCPU/GPU* umístěny do sdílené složky *OfficeSources*, spolu s GPU kernelem a jejich předpona vhodně přejmenována na *OfficeRC4Cracker*. Následně byl ve sdílené složce vytvořen hlavičkový soubor *EncryptionStructures.h*, do kterého byly přesunuty struktury pro ukládání šifrovacích informací či verze. Ty budou také použity napříč jednotlivým binárními Office formáty. Poté jsme již mohli přejít na úpravu samotného modulu DOC.

To, zda se jedná o staré RC4 šifrování, či novější CryptoAPI verzi, se určí podle kombinace hodnot verzí *vMajor* a *vMinor*, jak bylo popsáno v návrhu. Poté následuje buď klasický nebo CryptoAPI *EncryptionHeader*. Ten je stejný napříč formáty DOC, XLS i PPT, proto je zpracování této struktury opět prováděno ve sdílené složce *OfficeSources* v souboru *EncryptionHeaders.cpp*. Zpracování struktury *EncryptionHeader* pro starou metodu RC4 je již implementováno v původním modulu DOC a jedná se o pouhé přečtení tří hodnot *Salt*, *EncryptionVerifier* a *EncryptionVerifierHash* a uložení do struktury zasílané crackeru.

Verze CryptoAPI *EncryptionHeaderu* je mnohem obsáhlejší a kromě čtení hodnot bylo

prováděno i ověřování, zda se zde nachází přípustné hodnoty. Nejprve jsou ověřeny příznaky, tedy zda se opravdu jedná o CryptoAPI hlavičku a pro zašifrování nebyl použit externí poskytovatel. Dále samotná hlavička, ve které musí být nastaven šifrovací algoritmus na RC4 a hešovací funkce na SHA-1. Poté je přeskočen název poskytovatele šifrování, který se zde nachází v čitelné podobě (obvykle se jedná o *„Microsoft Enhanced Cryptographic Provider“*). Nakonec získáme potřebnou trojici hodnot, která se společně s konstantami jako je *SaltSize* nebo *VerifierHashSize* nachází ve struktuře *Verifier* na konci *EncryptionHeaderu*. Pokud vše proběhne v pořádku, jsou data uložena do struktury zasílané crackeru a úspěch inicializace je indikován nastavením příznaku *is_supported* na hodnotu `true`.

Implementace crackeru pro CryptoAPI verze dokumentů Office se nachází v souborech *OfficeCryptoAPIRC4CrackerCPU/GPU*. Pro potřeby tohoto crackeru byla implementována hešovací funkce SHA-1, a byla optimalizována pro co nejrychlejší vykonávání, například použitím maker, předpočítané tabulky hodnot nebo ručním vypsáním náročných cyklů. Implementaci této funkce se nachází v souboru *sha1.cpp*. Využívá se funkce *sha1_fast*, která nebere v úvahu vstupy delší než jeden blok, což je 64 bytů. To si můžeme dovolit, protože v binárních formátech dokumentů Office je maximální délka hesla 15 znaků. To je 30 bytů v kódování UTF-16. Pokud k tomuto přičteme dalších 16 bytů hodnoty *Salt*, která se zasílá hešovací funkci společně s heslem, stále tento řetězec splňuje limit jednoho 64-bytového bloku. Proudová šifra RC4 byla převzata ze staršího modulu DOC. Vytvoření šifrovacího klíče a následné ověření správnosti hesla tak, jak je popsáno v kapitole 6.1.1, se v CPU crackeru nachází ve funkci *checkPassword*.

Co se týče GPU verze crackeru, byl vytvořen kernel *office_cryptoapirc4_kernel.cl*. Do tohoto bylo nutné přepsat algoritmy SHA-1, RC4 a funkci *checkPassword* v jazyku OpenCL. Pozornost zde byla věnována například nemožnosti alokovat paměťové zdroje dynamicky. Také bylo využito speciálních formátů tohoto jazyka, jako jsou například vektory různých typů o fixně dané velikosti.

7.2 Modul XLS

Jedná se o zcela nový modul, který se v původním nástroji Wrathion nevyskytoval. Proto bylo nutné začít vytvořením souborů *XLSModule.cpp* a *XLSFormat.cpp*, ve kterých byly specifikovány základy, jako je přípona formátu a jeho signatura. Dále bylo nutné zahrnout tyto soubory do kompilace tak, aby je nástroj rozpoznal jako další modul.

Získávání údajů potřebných pro obnovu hesla se nachází ve funkci *init* a bylo zde postupováno dle návrhu v sekci 5.4. Po nalezení streamu *Workbook* v prvním directory sektoru začneme číst začátek datového sektoru tohoto streamu. Zde se totiž nachází substream *Globals*. V něm budeme hledat začátek záznamu *FilePass*, obsahující námi hledané informace. Ten je identifikovatelný svou hlavičkou o hodnotě `0x2F00`. Navíc by měl, v závislosti na dalších volitelných záznamech, ležet na prvních několika bytech substreamu. Pokud tedy projdeme prvních 100 bytů a záznam nenalezneme, prohlásíme soubor za nešifrovaný. V opačném případě přečteme 4 byty obsahující informace o délce záznamu a to, zda soubor není šifrován XOR obfuskací. Nyní se nacházíme na struktuře *EncryptionHeader*, která je identická ve všech dalších binárních formátech dokumentů Office. Postup je tedy totožný, jako v kapitole 7.1 výše. Podle použité verze je šifrovací hlavička zpracována jako stará RC4 40-bitová verze nebo jako CryptoAPI verze. Následně je použit jeden ze dvou stejnojmenných crackerů pro obnovení hesla.

7.3 Modul PPT

Opět se jedná o zcela nový modul nástroje Wrathion. Po specifikaci základních údajů v souborech *PPTModule.cpp* a *PPTFormat.cpp* bylo nutné získat šifrovací informace v podobě trojice hodnot – *Salt*, *EncryptedVerifier* a *EncryptedVerifierHash*. Toho je docíleno opět ve funkci *init*. Jak bylo popsáno v návrhu modulu v sekci 5.5, bude nás zajímat pouze tzv. *CryptSession10Container*, který je identifikovatelný posloupností bytů `0x0F00142F`. Jednoduše tedy procházíme binární data v dokumentu, dokud nenarazíme na posloupnost těchto dat. Následující 4 byty, informující o délce kontejneru, ignorujeme. Následuje struktura *EncryptionHeader*. U modulu PPT však nemusíme řešit, o kterou verzi se jedná, protože formát *.ppt* podporuje pouze CryptoAPI šifrování. Ve staré verzi 97-2000 totiž nelze dokument PowerPoint vůbec zašifrovat. Ověříme tedy pouze, zda se jedná o CryptoAPI verzi a hlavičku zpracujeme stejnou funkcí jako ostatní binární formáty Office. Rovněž cracker, respektive kernel pro ověření správnosti hesla, bude stejný.

7.4 Modul OOXML

Tento modul zahrnuje všechny formáty DOCX, XLSX a PPTX od verze 2007. Implementačně je však nutné rozdělit tento modul do tří různých modulů, podle přípony souborů. Tyto tři moduly, tedy DOCX, XLSX a PPTX budou inicializovány identicky, avšak každý samostatně. Naopak zpracování šifrovací hlavičky, společně s crackery, bude umístěno ve sdílené složce *OfficeSources* a bude využívány všemi třemi moduly.

Začátek inicializace modulů OOXML je velmi podobný tomu z předchozích modulů, protože základní informace o souboru jsou uloženy ve stejném binárním formátu. Po zjištění velikosti sektoru a pozice prvního datového sektoru se na něj přesuneme. Zde nalezneme stream *EncryptionInfo*, který obsahuje odkaz na strukturu *EncryptionHeader*. Zde nám první čtyři byty řeknou, zda se jedná o standardní šifrování (viz kapitolu 6.4.1) nebo o agilní šifrování (viz kapitolu 6.4.2). Podle toho je hlavička zpracována a je zavolán příslušný cracker.

7.4.1 Standradní šifrování

Standardní šifrování je použito, jak již bylo zmíněno, pouze ve verzi Office 2007. Informace o šifrování jsou zde, již naposled, uloženy v binární podobě. Struktura *EncryptionHeader* se velmi podobá CryptoAPI hlavičce, známé z binárních verzí. Jediný rozdíl je v příznacích a použitém šifrovacím algoritmu. Pokud proběhne zpracování hlavičky bez problémů, je zavolán cracker s názvem *AESStandardCrackerCPU*, respektive *AESStandardCrackerGPU*. V této verzi je šifrovací algoritmus RC4 nahrazen algoritmem AES. Ten je zde použit v režimu ECB a může mít libovolnou povolenou délku klíče tedy 128 bitů, 192 bitů nebo 256 bitů. Jeho implementaci je možné najít v souboru *aes.cpp* ve sdílené složce *OfficeSources*. Použitá hešovací funkce SHA-1 se nachází v souboru *sha1.cpp*. Nejedná se však o algoritmus identický s CryptoAPI verzí. Maximální délka hesla u verzí Office 2007 a novějších je totiž až 255 znaků. Musíme tedy počítat s nutností hešovat bloky delší než 64 bytů. Původní algoritmus byl tedy rozšířen a nachází se ve funkci *sha1*.

Získávání šifrovacího klíče a následné ověření hesla probíhá ve funkci crackeru *checkPassword* podle postupu rozebraném v sekci 6.4.1. Za zmínku stojí cyklus, který zajišťuje 50 000 iterací hešovací funkce SHA-1. Tento proces, jak se přesvědčíme při zkoumání experimentů v sekci 8.3, má drastický dopad na dobu, potřebnou pro obnovení hesla. Po získání

šifrovacího klíče je použita funkce *AES_ECB_decrypt* pro rozšifrování prvního bloku hodnot *EncryptedVerifier* a *EncryptedVerifierHash*. Dešifrovaná hodnota *Verifier* je následně zahešována a porovnána s dešifrovanou hodnotou *VerifierHash*. Při shodě obnova hesla úspěšně končí a cracker vrací odpovídající návratový kód.

Pro potřeby crackování s urychlením GPU byl vytvořen kernel *office_standardaes_kernel*, do kterého byl výše popsaný proces přepsán do jazyka OpenCL. Kromě již zmíněných problémů s nemožností alokace paměti dynamicky se zde vyskytl ještě další problém. Vzhledem k použití iterativního hešování se z procesu obnovy hesla stal výpočetně velmi náročný problém. Při využití standardu OpenCL se grafická karta věnuje plně zadanému úkolu a nereaguje na další žádosti. To má po několika okamžicích za důsledek obnovu grafického ovladače operačním systémem, čímž se přeruší proces obnovy hesla. Z tohoto důvodu je nutné při spouštění upravit adekvátně velikost pracovní skupiny, aby mohla grafická karta obsloužit zároveň hostovaný operační systém. Tato problematika, spolu s výběrem vhodné velikosti pracovní skupiny, bude blíže probrána při zkoumání experimentů v kapitole 8.

7.4.2 Agilní šifrování

Jedná se o poslední typ šifrování, který je použit nejnovějšími verzemi dokumentů Office, tedy 2010, 2013 a 2016. Inicializace proběhne stejně jako u standardního šifrování, nicméně způsob uložení informací ve struktuře *EncryptionHeader* je zcela odlišný. Nejedná se totiž o binární data, nýbrž o dokument formátu XML. Zpracování této struktury probíhá v souboru *EncryptionHeaders.cpp* ve funkci *XMLEncryptionHeaderParse*. Na uložení XML je předalokován buffer o velikosti 4kB. V době tvorby těchto modulů taková velikost bohatě dostačuje i na nejnovější verzi 2016.

XML dokument je zpracováván ručně, z důvodu omezení závislostí na externí knihovny. Informace, které budeme potřebovat, se nachází jako atributy v uzlu *p:encryptedKey*. Nejprve tedy extrahujeme řetězec obsahující pouze tyto atributy. Ty následně rozdělíme do dvojic a ve funkci *parseInfo* je uložíme do struktury následně zasílané crackerům.

CPU cracker agilní verze šifrování je nejrozsáhlejší cracker napříč různými verzemi modulů Office. Důvodem je možnost použití několika možných algoritmů pro hešování a šifrování. Ve verzi 2010 se standardně jedná o SHA-1 společně s AES-128 a od verze 2013 je to pak standardně SHA512 se šifrovacím algoritmem AES-256. Je nutno podotknout, že v agilní verzi je použit výhradně AES v režimu CBC. Standard dovoluje také režim CFB, ale ten není momentálně využívaný. Další algoritmy, které nástroj Wrathion podporuje v agilní verzi, jsou hešovací funkce SHA256, SHA384 a šifrovací algoritmus AES-192. Standard umožňuje použití i dalších algoritmů (viz kapitolu 6.4.2), tyto ale nejsou kvůli minimálnímu použití implementovány.

Cracker podle použité hešovací funkce zavolá při obnově každého hesla vhodnou funkci, která vytvoří šifrovací klíč. Tyto funkce jsou si velmi podobné, ovšem pro minimalizaci počtu rozhodování má každý hešovací algoritmus svoji funkci, tedy například *sha1_checkPassword*, *sha512_checkPassword*, apod. Na konci každé této funkce je volána metoda *decryptVerifiers*, které jsou předány získané šifrovací klíče. Tato metoda rozhodne, který šifrovací algoritmus se má použít při dešifrování kontrolních hodnot a provede tak finální rozhodnutí o shodě hesla.

Co se týče GPU verze, byly pro agilní šifrování vytvořeny dva crackery, z nichž každý používá odlišný šifrovací a hešovací algoritmus. To z důvodu, aby se minimalizovalo množství informací přenášených do GPU a následně rozhodování o vhodném algoritmu. Ve složce *kernels* se tedy nachází kernel *office_agileaes_sha1_kernel* pro metodu šifrování Office 2010,

tedy SHA1 a AES-128, a dále kernel *office_agileaes_sha512_kernel* pro verze Office 2013 a 2016. Zde je použita hešovací funkce SHA512 a šifrovací algoritmus AES-256. Opět zde byly použité algoritmy, spolu s algoritmem pro vygenerování šifrovacího klíče a ověření správnosti hesla, přepsány z jazyka C++ do OpenCL. Kromě přepisu dynamické práce s pamětí do odpovídajících konstrukcí se zde nenacházejí větší odlišnosti od CPU verze. Tou jedinou je u verze SHA512 ověřování pouze prvního bloku ověřovacích hodnot z důvodu problematické implementace AES v OpenCL. To by se teoreticky mohlo projevit generováním tzv. „false positive“ hesel, v praxi je však pravděpodobnost takového nálezu extrémně malá, tudíž zanedbatelná.

Kapitola 8

Experimenty

Pro účely provádění experimentů a měření je dobré brát v potaz jednotlivé crackery, nikoliv moduly či formáty. Výkonnost, tedy počet ověřených hesel za jednotku času, totiž záleží právě na použitém crackeru, nikoliv na modulu. V tom proběhne pouze inicializace, viz kapitolu 6 o návrhu nových modulů. Ve finální podobě tedy obsahuje nástroj Wrathion následující crackery pro obnovu hesel dokumentů Office, se kterými budou provedeny experimenty:

- **RC4 cracker** - Používaný nástroji Word 97-2000 a Excel 97-2000.
- **CryptoAPI cracker** - Používaný verzemi XP a 2003 nástrojů Word, Excel a PowerPoint, respektive vyššími verzemi těchto nástrojů, při uložení dokumentu ve formátu „Office 97-2003“.
- **Standardní cracker** - Používaný nástroji Word, Excel a PowerPoint verze 2007.
- **Agilní cracker** - Používaný nástroji Word, Excel a PowerPoint verzemi 2010 a novějšími. Navíc byly provedeny odlišné experimenty pro verzi 2010 a verze 2013 a 2016, kvůli použití odlišných algoritmů.

Pro měření výkonnosti nástroje, respektive jednotlivých crackerů, je nutné určit abecedu znaků, které mohou hesla obsahovat a také délku jednotlivých hesel. Počet všech možných hesel x o dané délce m, které se mohou skládat z k znaků, je možné vyjádřit pomocí vzorce 8.1.

$$x = k^m \tag{8.1}$$

Z toho vyplývá, že při zvětšování délky hesla bude čas potřebný pro jeho obnovu růst exponenciálně. Navíc, nástroj Wrathion nedovoluje zadat minimální délku hesla. Proto je pro obnovu hesla určité délky nutné ověřit i hesla všech předchozích délek. Tomu odpovídá vzorec 8.2.

$$x = \sum_{i=1}^{m} k^i \tag{8.2}$$

Pro experimenty bylo zvoleno fixní heslo, které obsahuje malé znaky anglické abecedy, tedy 26 možných znaků. Toto heslo je tvořeno znaky počínajíce začátkem abecedy a s každým dalším přírůstkem délky se o jeden znak posunují (tedy například „*abcde*“ pro 5 znakové heslo).

Samotné experimenty se pak zaměří jednak na měření čistého výkonu, tedy počtu hesel za sekundu, které dokáží jednotlivé crackery ověřit. Dále bude také pro představu doplněna tabulka, obsahující časy pro obnovu jednotlivých hesel. Délka těchto hesel byla stanovena v rozmezí 5-10 znaků, později však byla rozšířena ještě o délky 11 a 12 z důvodu vysokého výkonu crackerů starších verzí Office, tedy RC4 a CryptoAPI. Z opačného důvodu byla pro crackery verzí Office Open XML vyzkoušena také hesla délky 3 a 4. Při příliš dlouhé době, nutné pro obnovu hesla (v řádu hodin, dnů, atd.), byl použit skript, který s dostatečnou přesností odhaduje dobu potřebnou pro obnovení daného hesla. Použití tohoto skriptu je v tabulkách znázorněno modrou barvou.

Je nutné podotknout, že zvolená fixní hesla se nachází na začátku stavového prostoru dané délky. Proto by obnova většiny hesel identické délky mohla trvat déle, než bylo změřeno experimenty, a to v případě, že by generování probíhalo od počátku abecedy. Změřený čas je čas využitý samotným procesem obnovy hesel a nepočítá se do něj čas potřebný pro inicializaci. To však ve výsledku nemá výrazný vliv na konečný čas, protože se jedná o rozdíl maximálně několika sekund.

Měření bude prováděno jednak na CPU, ale také na grafických kartách. Bude měřen výkon jedné grafické karty a ten bude srovnán s výkonem při použití tří grafických karet. Předpoklad je takový, že by se výkon při použití 3 GPU měl ztrojnásobit. Musíme však také počítat se zvýšenou režií, která je spojená s distribucí hesel mezi více zařízení. Očekávaný výkon tak bude pravděpodobně o něco nižší. I když máme k dispozici 4 grafické karty, z důvodu eliminace vlivu první, pomalejší, grafické karty, která je zatížena zobrazením obrazu a nástrojem TeamViewer, byla využita pouze 3 zařízení GPU.

Dále také budou provedeny experimenty s konkurenčními nástroji pro obnovu hesel. Mezi ně jsou zařazeny nástroje oclHashcat, John the Ripper a Elcomsoft Advanced Office Password Recovery. Bude provedeno srovnání výkonu nástroje Wrathion, respektive jeho nových modulů, s těmito nástroji a získané výsledky budou podrobně rozebrány.

Experimenty budou prováděny vzdáleně, pomocí nástroje TeamViewer, na testovacím stroji s následujícími komponentami:

- **Základní deska** - ASUS ROG RAMPAGE V EXTREME,
- **procesor** - Intel Core i7-5930K,
- **grafická karta** - 4x AMD R9 Fury X,
- **paměť** - Kingston 32GB 2800MHz HyperX Predator,
- **operační systém** - Windows 8.1.

Použitý procesor má 6 jader a při obnově bude využito všech 12 vláken. Bude se tedy také jednat o určitý stupeň paralelizace. U obnovy s využitím grafických karet bude záležet na zvolené velikosti pracovní skupiny. Pokud zanedbáme režii, která se negativně projeví pouze u kratších hesel, můžeme uvažovat s rostoucí velikostí skupiny také rostoucí výkon. To ovšem nebude platit u kernelů pro OOXML formáty, které jsou výpočetně velmi náročné a pro větší pracovní skupiny způsobují obnovení ovladače grafického adaptéru. Proto byla při experimentech volena vhodná velikost této skupiny. Tento proces bude blíže popsán v odpovídajících sekcích níže.

8.1 RC4 cracker

Tento cracker je používaný starým modulem DOC, to znamená, že se vyskytoval již v původní verzi nástroje Wrathion. Nicméně ho využívá i starší verze 97-2000 dokumentů Excel. Navíc díky tomu, že jsme s tímto crackerem provedli experimenty, můžeme porovnat rychlosti obnovy hesel u staré verze Office 97-2000 a novější CryptoAPI verze. Naměřené časy obnovy hesel různých délek je možné vidět v tabulce B.1. Porovnání čistého výkonu, tedy počtu vyzkoušených hesel za sekundu, je znázorněno v grafu C.1.

Z tabulky lze vyčíst, že i 10 znakové hesla je možné bez problémů obnovit do 24 hodin. Z toho plyne velké bezpečnostní riziko použití metody zabezpečení dokumentů této staré verze nástrojů Office Word a Excel.

Z grafu vyplývá, že zrychlení GPU oproti procesoru je přibližně čtyřnásobné. Pokud bychom však vzali v úvahu práci pouze v jednom vlákně, je zrychlení téměř 50-násobné. S tímto údajem se dostáváme blízko měření, které bylo provedeno s původním modulem DOC. Zrychlení je v tomto případě větší, protože byla použita výkonnější grafická karta.

Co se týče měření se třemi grafickými kartami, je zde zrychlení oproti jedné kartě téměř přesně trojnásobné. Tím se splnilo naše očekávání o použití více grafických karet.

8.2 CryptoAPI cracker

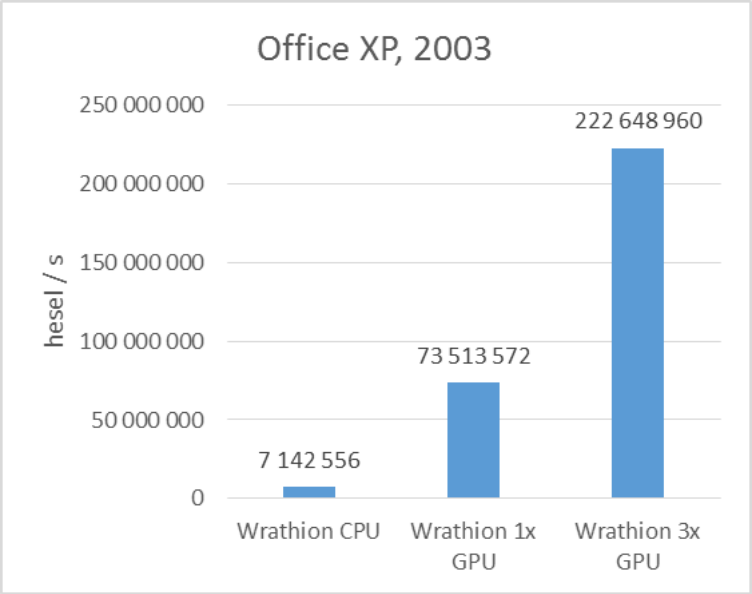
Jedná se o první cracker, který byl v rámci této práce vytvořen. Zajišťuje podporu verzí XP a 2003 dokumentů Office Word a Excel a nově také nástroje Office PowerPoint, který nebylo možné v předchozích verzích šifrovat. I když se jedná o novější formát a byla použita novější hešovací funkce SHA-1, z tabulky B.2 si je možné všimnout, že časy nutné pro obnovu hesla jsou ještě nižší, než ve starší verzi. To opět svědčí o dnes již nedostatečné spolehlivosti tohoto zabezpečení. Důvody takovýchto výsledku budou rozebrány níže při porovnávání jednotlivých verzí v sekci 8.5.

V grafu C.2 jsou zobrazeny rozdíly mezi použitím CPU, GPU a více GPU současně. Je zde možné vidět, že obnova na GPU je více než 10x rychlejší než obnova na 12 vláknech CPU. Tedy téměř 125-násobné zrychlení oproti sekvenčnímu zpracování v jednom vlákně. U tohoto crackeru je tedy vidět opravdový potenciál použití grafických karet pro obnovu hesla. Při užití 3 zařízení GPU můžeme sledovat opět trojnásobné zrychlení výkonu oproti jedné GPU tak, jak bylo očekáváno. To je také možné vidět v grafu 8.1.

8.3 Standradní AES

Pro standardní verzi šifrování, tedy verzi Office 2007, byl vytvořen samostatný cracker. Jak je vidět na časech uvedených v tabulce B.3 i v grafu C.3 porovnávajícím různé možnosti obnovy, rychlost obnovy u tohoto formátu rapidně klesla. To je mimo jiné způsobeno zavedením iterativního hešování, které je nutné při tvorbě šifrovacího klíče. V této verzi je počet iterací pevně stanoven na 50 000. V budoucích verzích se však zpomalení projeví ještě více, což způsobí kromě zvyšování počtu iterací i zavedení novější hešovací funkce SHA512. Také byl v této verzi poprvé nahrazen šifrovací algoritmus RC4 algoritmem AES. Ten je zde standardně ve 128-bitové verzi, vykonáván v režimu ECB.

Z předchozích závěrů o nespolehlivosti starých metod zabezpečení se jedná o logický krok společnosti Microsoft. Z výsledků grafů můžeme usoudit, že se jednalo o úspěšný krok,



Obrázek 8.1: Porovnání CPU a GPU obnovy hesla verze Office XP a 2003

protože doba potřebná pro obnovu hesla u této verze vzrostla asi 7700 krát oproti staré verzi 2003. O to více se zde projevuje výhoda použití paralelizace za pomocí GPU.

Bohužel, kvůli náročnosti algoritmu tvorby šifrovacího klíče se nepovedlo využít plného potenciálu grafické karty. Při experimentech musela být snižována velikost pracovní skupiny z původních 350 000, použitých pro měření verzí 97-2003, na pouhých 20 000. Toto číslo se jevilo jako nejvíce stabilní volba pro experimenty. I tak však, dle grafu výše, dosahovala grafická karta více než 13-násobného zrychlení oproti 12 vláknovému procesoru, tedy proces obnovy hesla zvládala více než 150x rychleji nežli sekvenční provádění.

8.4 Agilní AES

Tento cracker se používá u nejnovějších verzí Office, tedy 2010, 2013 a 2016. I když proces generování hesla je u těchto verzí stejný, ve verzi 2010 je použit jiný hešovací a šifrovací algoritmus, než v posledních dvou verzích. Ze stejného důvodu byly také pro GPU vytvořeny dva různé kernely, viz kapitolu 7.4.1. Proto bylo provedeno zvlášť měření pro verzi 2010 a verze následující.

8.4.1 Verze Office 2010

Tato verze standardně používá pro tvorbu šifrovacího klíče 100 000 iterací hešovací funkce SHA-1 a 128-bitový šifrovací algoritmus AES v režimu CBC. Došlo tedy, kromě změny algoritmu pro tvorbu šifrovacího klíče, také ke zdvojnásobení standardního počtu iterací hešovací funkce. Z tabulky B.4 je zřejmé, že to mělo za následek zpomalení procesu obnovy na poloviční rychlost oproti verzi standardního šifrování.

v grafu C.4 je vidět, že množství vyzkoušených hesel za sekundu kleslo na řád několika stovek. Pokud bychom uvažovali sekvenční zpracování v jednom vlákně, dokonce několika desítek. Díky paralelnímu zpracování na grafické kartě můžeme dosáhnout opět téměř 150-násobného zrychlení a to i v případě užití malé pracovní skupiny, která byla volena stejně,

jako u standardního šifrování. Co se týče zpracování na 3 zařízeních GPU, dostáváme se opět téměř na trojnásobek původní rychlosti. Tentokrát je číslo o něco nižší, což může být způsobeno právě velkým vytížením jednotlivých pracovních jednotek.

8.4.2 Verze Office 2013 a 2016

Poslední verze nástrojů Office, tedy 2013 a 2016, jsou totožné jak v algoritmu pro tvorbu šifrovacího klíče, tak v použitém hešovacím a šifrovacím algoritmu. Jmenovitě se jedná o SHA512 a 256-bitový AES použitý opět v režimu CBC. I když počet iterací zůstal stejný, zavedení nové hešovací funkce SHA512, patřící do nového standardu SHA-2, opět výrazně zpomalilo proces obnovy hesla dokumentů Office této verze. To je možné vidět i z naměřených časů v tabulce B.5 a změřených rychlostí v grafu C.5. Jedná se tedy o nejvíce zabezpečené verze z rodiny dokumentů Office.

Z tabulky je možné vyvodit, že obnova náročnějších hesel již nebude prakticky uskutečnitelná klasickou metodou brute-force. Řešením může být použití slovníkových útoků, či generování hesel dle jistých heuristik. Z grafu vyplývá opět velmi dobré zrychlení při použití grafické karty oproti CPU a to i s omezenou velikostí pracovní skupiny. Při užití třech grafických karet jsme dostali více než trojnásobné zrychlení procesu obnovy. Tím jsme opět prokázali pravdivost našeho předpokladu o násobení rychlosti obnovy při užití více zařízení GPU.

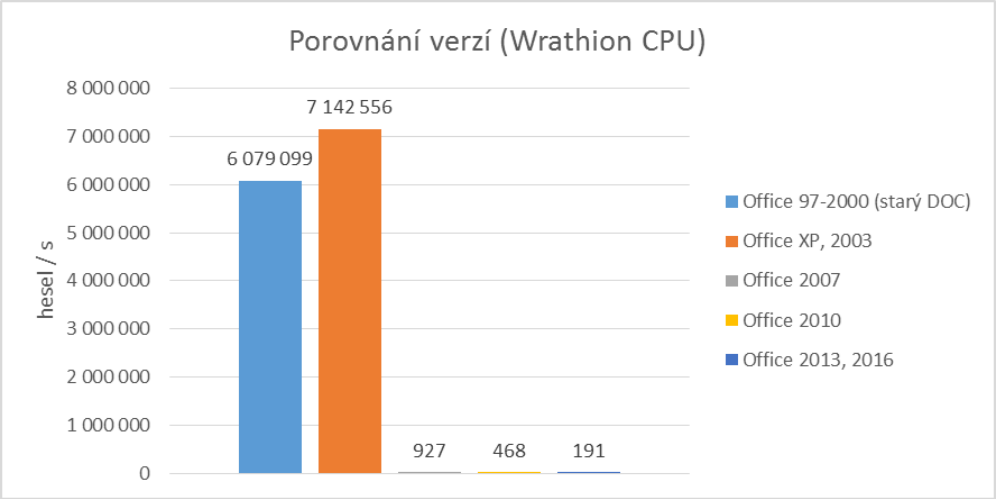
8.5 Srovnání jednotlivých verzí Office

Srovnáním výkonu našich crackerů můžeme získat dobrou představu o míře zabezpečení jednotlivých verzí dokumentů Office. Toto srovnání je vyneseno ve třech grafech. V grafu D.1 je porovnání obnovy hesel pomocí CPU a v grafu D.3, respektive D.5, je vidět porovnání jednotlivých verzí Office při obnově hesel s použitím jednoho, respektive tří zařízení GPU.

Pokud bychom porovnávali starý Office 97-2000 s CryptoAPI verzí použitou ve verzích Office XP a 2003, zjistili bychom, že novější verze nám dovoluje heslo obnovit rychleji. V CryptoAPI verzi byla sice zavedena nová hešovací funkce SHA-1, ta se ale rychlostí blíží starší hešovací funkci MD5, která je již dnes považována za prolomenou. Důvodem, proč je novější verze rychlejší, je algoritmus pro tvorbu šifrovacího klíče. Ve staré verzi je prováděno hešování celkem 4x, z toho jeden vstup má velikost 336 bytů. To znamená, že pro tento vstup musí být prováděno několik kol hešovací funkce. V novější verzi je SHA-1 prováděno pouze 3x a to při maximální velikosti 1 bloku. To je možné zajistit díky maximální délce hesla 15 znaků. Tedy i po rozšíření do UTF-16 a přidání 20 bytové hodnoty *Salt* se do tohoto 64-bytového limitu vlezeme. U GPU verzí je tento rozdíl ještě větší, což může být způsobeno lepší optimalizací CryptoAPI kernelu.

Při pohledu na grafy je vidět propastný rozdíl mezi původními binárními a OOXML formáty, zavedenými s verzí Office 2007. Z toho důvodu je také v grafech D.2, D.4, repektive D.6, zobrazen detail OOXML verzí v režimech CPU, GPU, respektive 3x GPU. Jak již bylo zmíněno, u verze 2007 je tento rozdíl způsoben zavedením iterativního hešování s pevně stanoveným počtem 50 000 cyklů. S verzí 2010 se tento počet zvýšil standardně na 100 000. Verze 2013 a 2016 pak přišly s novou verzí hešovací funkce, SHA512, což mělo za následek další zpomalení procesu obnovy hesla.

Z grafů je také zřejmá vzájemná podobnost, která se odráží právě v míře zabezpečení dané verze Office – tedy čím menší počet vyzkoušených hesel za jednotku času, tím náročnější je heslo u dané verze obnovit. Příklad CPU verze je vidět v grafu 8.2.



Obrázek 8.2: Srovnání jednotlivých verzí Office – CPU

8.6 Srovnání nástroje Wrathion s konkurenčními nástroji

Nástroj Wrathion byl porovnán se třemi nástroji, používanými pro obnovu hesel dokumentů Office – oclHascat, John the Ripper a nástrojem od společnosti Elcomsoft, Advanced Office Password Recovery. V následující kapitole budou shrnuty získané výsledky a budou vyneseny do grafu. Pro každou verzi dokumentů Office, respektive pro každý cracker, byl vytvořen graf, pro který jsou porovnány všechny 4 testované nástroje a to ve 3 režimech – CPU, 1x GPU a 3x GPU, tedy obnova hesla pomocí CPU, jedné grafické karty a 3 zařízení GPU. Je nutné podotknout, že konkurenční nástroje mnohé z těchto verzí, respektive variant, nepodporují. Proto je místo v grafu a přidružené tabulce prázdné. Tyto detaily budou probrány v podsekcích níže. Grafy je možné vidět na obrázcích E.1 – verze 97-2000, E.2 – verze 2003, E.3 – verze 2007, E.4 – verze 2010 a nakonec E.5 – verze 2013 a 2016.

8.6.1 Nástroj oclHashcat

Tento nástroj pro obnovu hesel, jak již napovídá název, používá pro urychlení procesu obnovy standardu OpenCL. Tento software přešel v průběhu vypracovávání této práce do podoby Open Source a na oficiálních stránkách se chlubí nejrychlejším procesem obnovy hesel na světě. Byl tedy velmi vhodným kandidátem pro porovnání s nástrojem Wrathion. Původně bylo v plánu porovnat i CPU variantu obnovy hesla, později však bylo zjištěno, že CPU Hashcat dokumenty Office nepodporuje. Pro experimenty byla použita verze 2.01.

Nástroj oclHashcat v čistém výkonu předehnal GPU obnovu hesel nástroje Wrathion ve všech verzích. Největší rozdíl je u OOXML formátů a to kvůli nutnosti zmenšování pracovní skupiny při provádění experimentů s nástrojem Wrathion. Proto by byla vhodná optimalizace OpenCL kernelů, aby bylo možné plně využít potenciálu grafických karet. V takovém případě by se mohl nástroj Wrathion nástroji oclHashcat vyrovnat.

Rychlost uváděná v grafu je však optimální rychlost nástroje oclHashcat. Tento nástroj, jak se ukázalo z experimentů, rychlost škáluje podle velikosti prozkoumávaného stavového prostoru. V praxi pak tedy uvedené rychlosti dosáhne při obnově delších hesel. Při menších délkách (například 6 a méně) je nástroj Wrathion rychlejší. Pro porovnání, obnova 3-znakového hesla dokumentu Office trvala nástroji Wrathion 30 sekund, zatímco oclHa-

shcat tuto činnost prováděl skoro 3 minuty. Samotné vyzkoušení prvních 26 znaků abecedy nástroji oclHashcat zabralo přes minutu, tedy potřeboval více než 2 sekundy na ověření jediného hesla.

Další překážkou, kvůli které jsme nemohli porovnat absolutní časy potřebné pro obnovu určitého hesla, byl způsob generování hesel. Zatímco nástroj Wrathion generuje hesla postupně, od začátku abecedy, nástroj oclHashcat používá modifikované Markovovy řetězce. Proto námi vybraná fixní hesla byla obnovena rychleji nástrojem Wrathion nežli nástrojem oclHashcat.

8.6.2 Nástroj John the Ripper

Jedná se taktéž o Open Source software pro obnovu hesel dokumentů nejrůznějších formátů. Pro experimenty byla použita verze John the Ripper 1.8.0 Jumbo 1.

Na rozdíl od oclHascat tento nástroj podporuje i obnovu hesel pomocí CPU. Bohužel se však nepodařilo zprovoznit obnovu hesel na více zařízeních GPU. I když takovou variantu tento nástroj oficiálně podporuje, při správném požadavku na použití více grafických karet tento nástroj stále využíval pouze jedinou. Pravděpodobně se tedy jednalo o chybu daného software.

Pokud porovnáme CPU řešení tohoto nástroje s nástrojem Wrathion, dosahuje John the Ripper srovnatelného výkonu. U starších formátů Office je nástroj Wrathion dokonce rychlejší a to o jednu třetinu u verze 97-2000 a jednu šestinu u CryptoAPI verze. I v nejnovější verzi Office dosahuje John the Ripper asi jen 80% výkonu nástroje Wrathion.

Při srovnání rychlosti procesu obnovy hesla s použitím grafické karty je nástroj Wrathion u binárních verzí Office rychlejší, u CryptoAPI verze až 3x. U OOXML verzí se však opět projevila omezená velikost pracovní skupiny, což způsobilo, že John the Ripper je až 8x rychlejší. V těchto verzích je dokonce výkonnější než nástroj oclHashcat. Zajímavé je však totální selhání nástroje John the Ripper u GPU obnovy hesel verze 2013 a 2016. V těchto verzích dosahuje výkonu asi 21 vyzkoušených hesel za sekundu, což je téměř 60x méně než nástroj Wrathion.

8.6.3 Nástroj Elcomsoft AOPR

Jedná se o komerční software, jehož licence byla poskytnuta fakultou. Experimenty s tímto nástrojem mohly být provedeny pouze v režimu CPU. Při pokusu o využití jedné či více grafických karet se buď vyskytla chybová hláška o selhání vnitřní funkce tohoto software, nebo nástroj požadavek ignoroval a obnovu prováděl na zařízení CPU. Jak bylo později zjištěno, toto chování způsobovala pravděpodobně nová architektura Fiji, na které je postavena grafická karta AMD R9 Fury X, která se nacházela v testovacím stroji. Při použití starších grafických karet se nástroj choval korektně. Jednoznačně se tedy jednalo o chybu software, která by při správném použití standardu OpenCL neměla nastat. Výsledky získané na jiném testovacím stroji by však neměli vzhledem k ostatním naměřeným údajům žádný smysl, proto je zde neuvádím. V grafech jsou tedy vyneseny údaje pouze pro variantu CPU.

U binárních verzí Office nástroj společnosti Elcomsoft dosahoval o něco horších výsledků než nástroj Wrathion i John the Ripper. U OOXML verzí však pokořil tento nástroj oba konkurenty a dosáhl téměř trojnásobné rychlosti oproti nástroji Wrathion. Výjimkou byla opět nejnovější verze 2013 a 2016, ve které dosáhl Wrathion dvojnásobného zrychlení oproti nástroji od Elcomsoft.

Kapitola 9

Závěr

V rámci této práce byl proveden rozbor formátů Microsoft Office, jmenovitě formátů DOC a DOCX pro nástroj MS Word, dále XLS a XLSX pro nástroj MS Excel a formátů PPT a PPTX pro nástroj MS PowerPoint. Analyzovali jsme jejich strukturu a získali informace potřebné pro úspěšnou obnovu hesla zašifrovaných dokumentů zmíněných formátů. Ty zahrnují informace o použitém šifrovacím a hešovacícm algoritmu, ověřovací hodnoty pro zjištění správnosti hesla, ale také informace o verzi dokumentu Office. Ta je pro nás velmi důležitá, protože struktura dokumentů se liší téměř s každou novou verzí. Tou největší změnou je způsob uložení ověřovacích hodnot nutných k úspěšné obnově hesla. Do verze Office 2007 se používá binární formát napodobující souborový systém, od verze 2010 se pak šifrovací informace uchovávají ve formátu XML.

Všechny výše zmíněné informace, společně se znalostmi o hešovacíh a šifrovacích algoritmech, jsme využili při návrhu přídavných modulů pro nástroj Wrathion, které rozšiřují jeho schopnosti o obnovu hesel výše zmíněných formátů.

Následovala implementační část, ve které byly úspěšně implementovány všechny navržené moduly. Díky tomu nástroj Wrathion nyní podporuje obnovu hesel všech nástrojů kancelářské řady Office, které je možné zašifrovat a to ve všech verzích od Office 97-2000 do nejnovější Office 2016 včetně. Byla implementována jak klasická verze pro obnovu hesla pomocí CPU, tak verze vhodná pro paralelní zpracování pomocí grafické karty s využitím standardu OpenCL.

Po úspěšné implementaci byly provedeny experimenty, ve kterých byla srovnána rychlost obnovy hesla pomocí CPU a GPU. Také proběhly testy škálovatelnosti výkonu nástroje Wrathion při použití více zařízení GPU. Při těchto experimentech se potvrdilo, že při využití tří grafických karet se výkon nástroje ztrojnásobí. Můžeme tedy očekávat podobný nárůst při použití více zařízení GPU či dokonce při využití distribuovaného řešení. V neposlední řadě byl nástroj Wrathion srovnán se světovou konkurencí v oblasti obnovy hesel, jak Open Source, tak komerční, a výsledky byly vyneseny do tabulek a grafů.

Pomocí experimentů jsme ukázali potenciál grafických karet pro paralelizaci a tedy masivní urychlení celého procesu obnovy hesla dokumentů Office i dalších formátů.

9.1 Výhled do budoucna

Při implementaci OpenCL kernelů pro nejnovější formáty dokumentů Office se nepodařilo plně využít potenciálu grafické karty, kvůli velké výpočetní náročnosti na jednotlivé pracovní jednotky. Proto by v budoucnu byla vhodná optimalizace těchto kernelů, která by

dovolovala využití větších pracovních skupin v rámci celého zařízení. Díky tomu bychom mohli dosáhnout dalšího zrychlení při zpracování na GPU.

Dále experimenty ukázaly, že brute-force útoky na nejnovější verze dokumentů Office již nedosahují dostatečného výkonu, aby byly schopny obnovit i středně složitá hesla v rozumném čase. Proto by bylo vhodné do nástroje Wrathion přidat inteligentní generování hesel například pomocí Markovových řetězců nebo zadáním masky hledaného hesla.

Vzhledem k tomu, že při vypracovávání tohoto projektu rozšiřovalo nástroj Wrathion několik osob současně, bylo by také vhodné všechny tyto změny shrnout a provést restrukturalizaci celého nástroje. Při obnově různých formátů se používají často identické hešovací algoritmy. Bylo by tedy logické tyto algoritmy umístit do jisté sdílené složky, odkud by je mohl používat jakýkoliv modul, například v podobě dynamické knihovny, bez potřeby opětovné implementace. Tedy tak, jak to již nyní funguje v současných modulech pro podporu obnovy hesel dokumentů Office.

Literatura

[1] Office Open XML File Formats – Fundamentals and Markup Language Reference. Technická zpráva, Ecma International, 2012.

[2] [MS-CFB]: Compound File Binary File Format. Technická zpráva, Microsoft, 2015, https://msdn.microsoft.com/en-us/library/dd942138.aspx.

[3] [MS-DOC]: Word (.doc) Binary File Format. Technická zpráva, Microsoft, 2015, https://msdn.microsoft.com/en-us/library/cc313153.aspx.

[4] [MS-OFFCRYPTO]: Office Document Cryptography Structure. Technická zpráva, Microsoft, 2015, https://msdn.microsoft.com/en-us/library/cc313071.aspx.

[5] [MS-PPT]: PowerPoint (.ppt) Binary File Format. Technická zpráva, Microsoft, 2015, https://msdn.microsoft.com/en-us/library/cc313106.aspx.

[6] [MS-XLS]: Excel Binary File Format (.xls) Structure. Technická zpráva, Microsoft, 2015, https://msdn.microsoft.com/en-us/library/cc313154.aspx.

[7] Bogdanov, A.; Khovratovich, D.; Rechberger, C.: Biclique cryptanalysis of the full AES. *Advances in Cryptology – ASIACRYPT*, 2011: s. 344–371.

[8] Howes, L.; Munshi, A.: The OpenCL Specification [online]. https://www.khronos.org/registry/cl/specs/opencl-2.1.pdf, 29. 1. 2015 [cit. 13. 11. 2015].

[9] Howes, L.; Munshi, A.: The OpenCL Extension Specification [online]. https://www.khronos.org/registry/cl/specs/opencl-2.0-extensions.pdf, 29. 1. 2015 [cit. 31. 11. 2015].

[10] Hranický, R.; Matoušek, P.; Veselý, V.; aj.: Experimental Evaluation of Password Recovery in Encrypted Documents. In *Proceedings of ICISSP 2016*, Roma: SciTePress - Science and Technology Publications, 2016, ISBN 978-989-758-167-0, s. 299–306.

[11] Menezes, A. J.; Vanstone, S. A.; Oorschot, P. C. V.: *Handbook of applied cryptography*. CRC Press, Inc. Boca Raton, 1996, ISBN 0849385237.

[12] Popov, A.: Prohibiting RC4 Cipher Suites. Technická zpráva, Microsoft Corp., 2015.

[13] Rivest, R.: The MD5 message-digest algorithm. Technická zpráva, MIT Laboratory for Computer Science and RSA Data Security, Inc., 1992.

[14] Schmeh, K.: *Cryptography and public key infrastructure on the Internet*. John Wiley & Sons, 2003, ISBN 978-0-470-84745-9, 45 s.

[15] Schmied, J.: *GPU akcelerované prolamování šifer*. Diplomová práce, FIT VUT v Brně, Brno, 2014.

[16] Schneier, B.: Cryptanalysis of SHA-1 [online]. https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html, 18. 2. 2005 [cit. 13. 12. 2015].

[17] Simmons, G. J.: Symmetric and asymmetric encryption. *ACM Computing Surveys (CSUR)*, 1979: s. 305–330.

[18] Wang, X.; Yu, H.: How to Break MD5 and Other Hash Functions. *Advances in Cryptology – EUROCRYPT*, 2005: s. 19–35.

[19] Wilson, B. F.: *Codes, Ciphers, Secrets and Cryptic Communication: Making and Breaking Sercet Messages from Hieroglyphocs to the Internet*. Black Dog & Leventhal Pub, 2005.

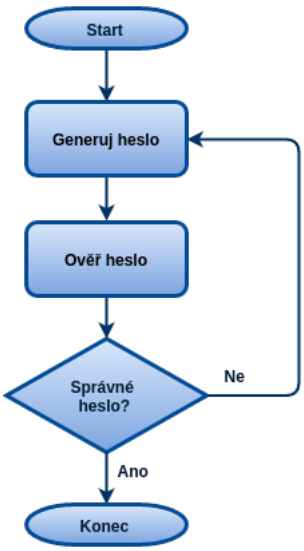
[20] Wu, X.; Hong, J.; Zhang, Y.: Analysis of OpenXML-based Office Encryption Mechanism. In *The 7th International Conference on Computer Science & Education*, 2012.

Přílohy

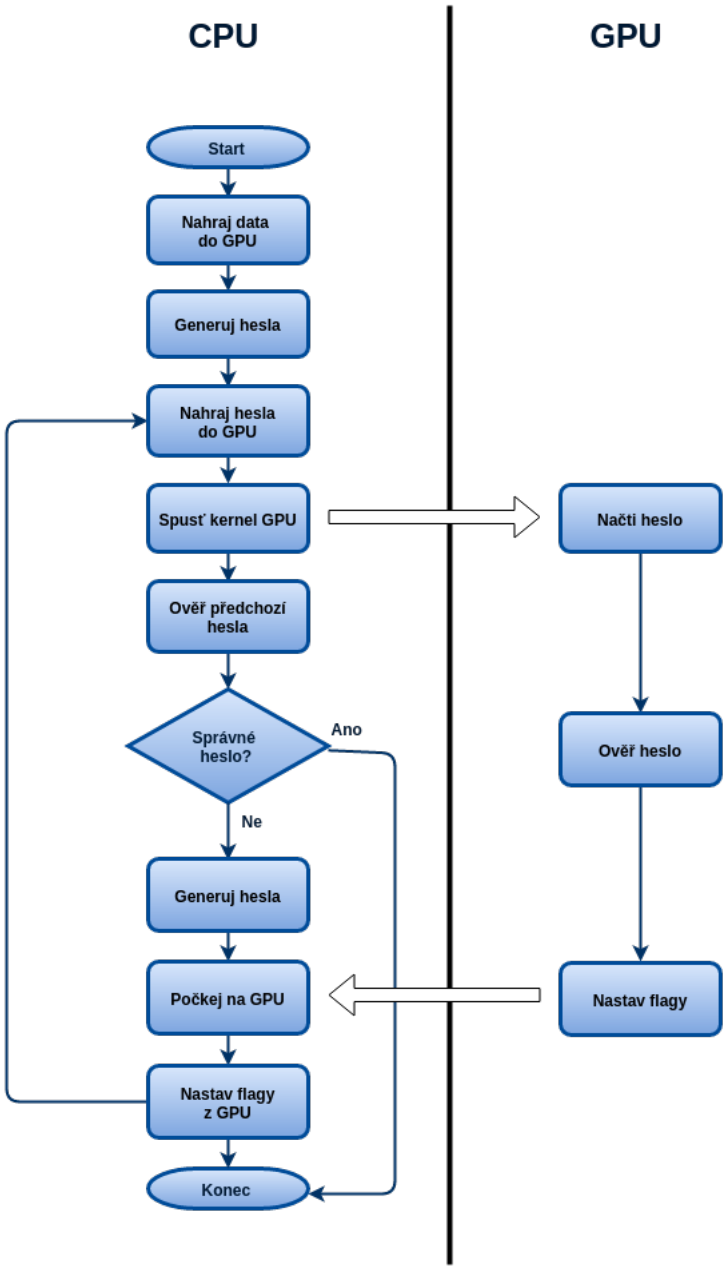
Seznam příloh

Příloha A

Schémata nástroje Wrathion



Obrázek A.1: Schéma prolamování hesel na CPU



Obrázek A.2: Schéma prolamování hesel na GPU

Příloha B

Tabulky časů obnovy hesel

Tabulka B.1: Doba prolomení hesla dokumentů Office verze 97-2000

délka hesla	5	6	7	8	9	10	11	12
CPU	1s	2,87s	55,06s	23m 55s	10h 19m	11d 4h	290d 16h	20y 258d
1x GPU	1,52s	1,52s	14,59s	5m 54s	2h 30m	2d 17h	70d 18h	5y 15d
3x GPU	1,52s	1,52s	6,54s	2m 0s	51m 9s	21h 46m	23d 14h	1y 248d

Tabulka B.2: Doba prolomení hesla dokumentů Office verze XP, 2003 a novějších

délka hesla	5	6	7	8	9	10	11	12
CPU	1s	1,85s	47,8s	20m 20s	8h 47m	9d 12h	247d 10h	17y 227d
1x GPU	1,52s	1,52s	5,54s	2m 0s	51m 48s	22h 11m	24d 0h	1y 260d
3x GPU	1,52s	1,52s	3,53s	40,73s	17m 14s	7h 19m	7d 22h	248d 8h

Tabulka B.3: Doba prolomení hesla dokumentů Office verze 2007

délka hesla	3	4	5	6	7	8	9
CPU	6,03s	20,39s	9m 22s	3h 51m	4d 4h	108d 11h	7y 265d
1x GPU	3,52s	5,54s	42,95s	16m 2s	6h 54m	7d 11h	194d 19h
3x GPU	4,53s	5,55s	18,67s	6m 1s	2h 34m	2d 19h	72d 17h

Tabulka B.4: Doba prolomení hesla dokumentů Office verze 2010

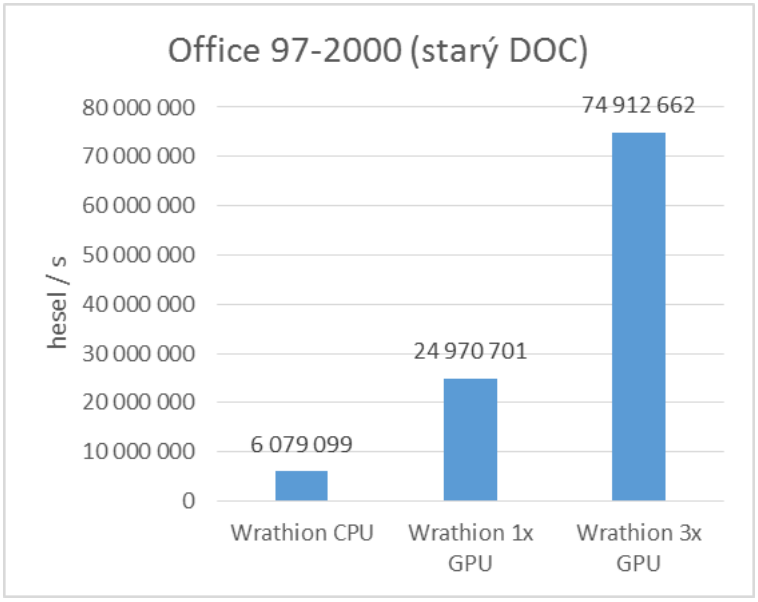
délka hesla	3	4	5	6	7	8	9
CPU	11,03s	43,14s	18m 17s	7h 37m	8d 6h	214d 20h	15y 110d
1x GPU	7,54s	10,57s	1m 41s	39m 24s	16h 37m	18d 8h	1y 103d
3x GPU	7,57s	10,60s	41,86s	15m 44s	6h 42m	7d 6h	188d 14h

Tabulka B.5: Doba prolomení hesla dokumentů Office verze 2013 a 2016

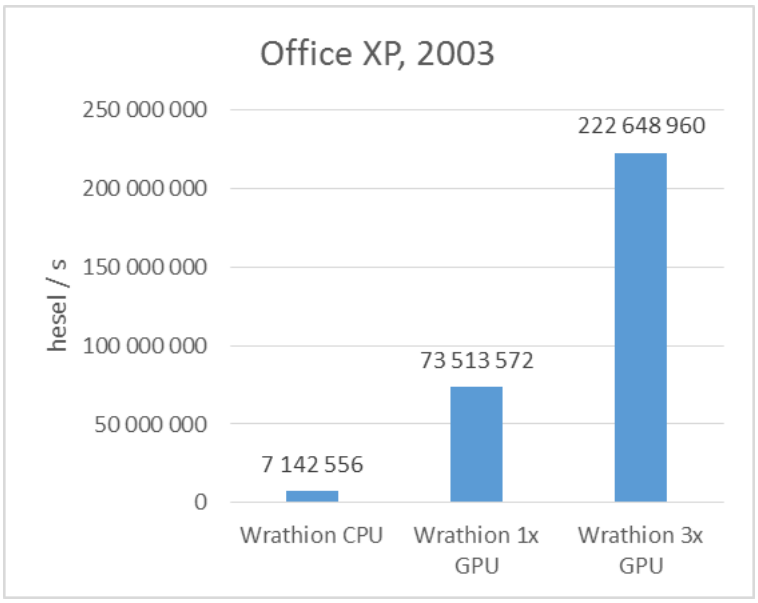
délka hesla	3	4	5	6	7	8	9
CPU	27,15s	1m 45s	43m 2s	18h 41m	20d 5h	1y 161d	37y 181d
1x GPU	31,66s	41,72s	6m 48s	2h 53m	3d 3h	81d 13h	5y 295d
3x GPU	31,41s	48,88s	2m 49s	53m 59s	23h 19m	25d 6h	1y 292d

Příloha C

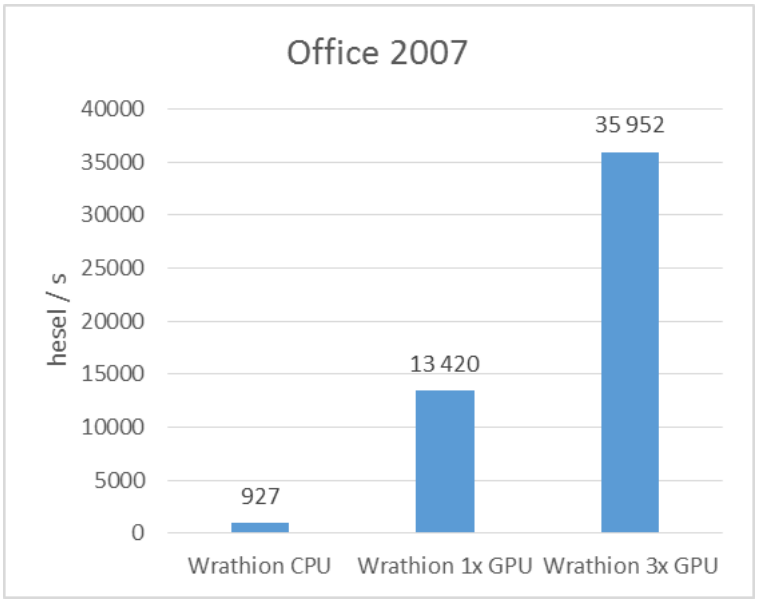
Grafy srovnání CPU a GPU obnovy hesel



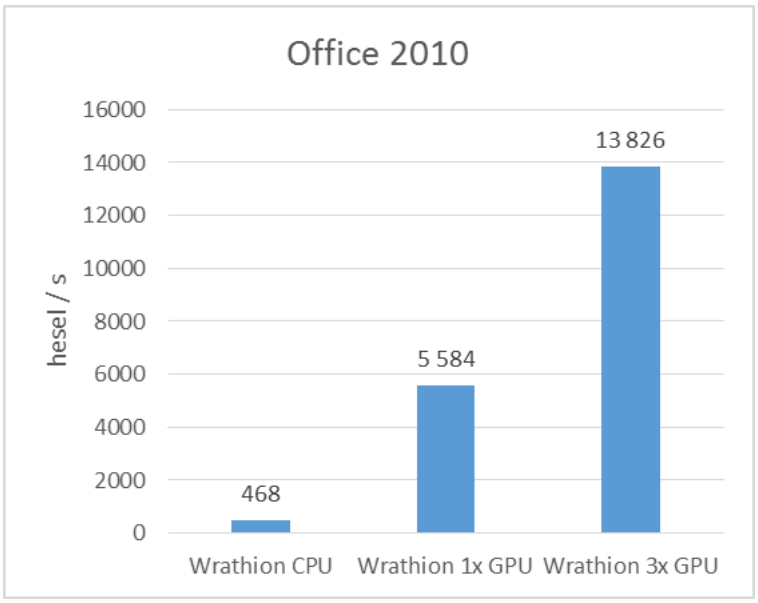
Obrázek C.1: Porovnání CPU a GPU obnovy hesla verze Office 97-2000



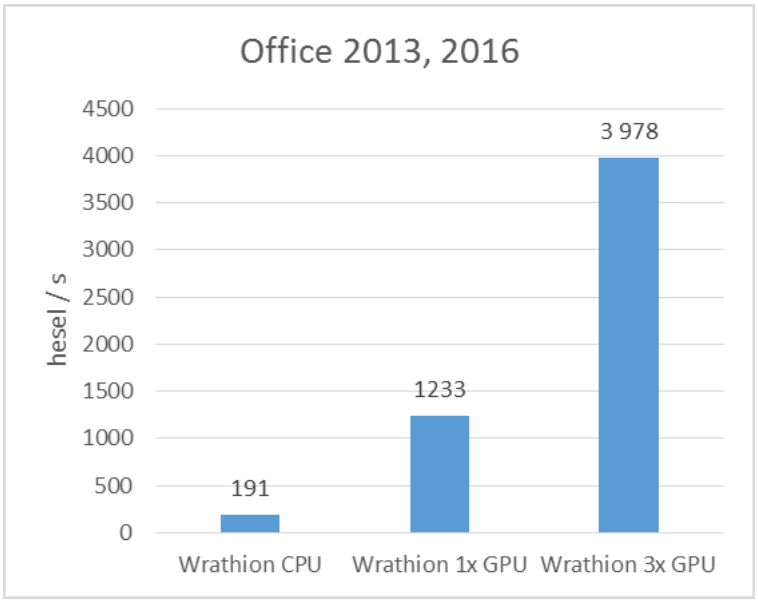
Obrázek C.2: Porovnání CPU a GPU obnovy hesla verze Office XP a 2003



Obrázek C.3: Porovnání CPU a GPU obnovy hesla verze Office 2007



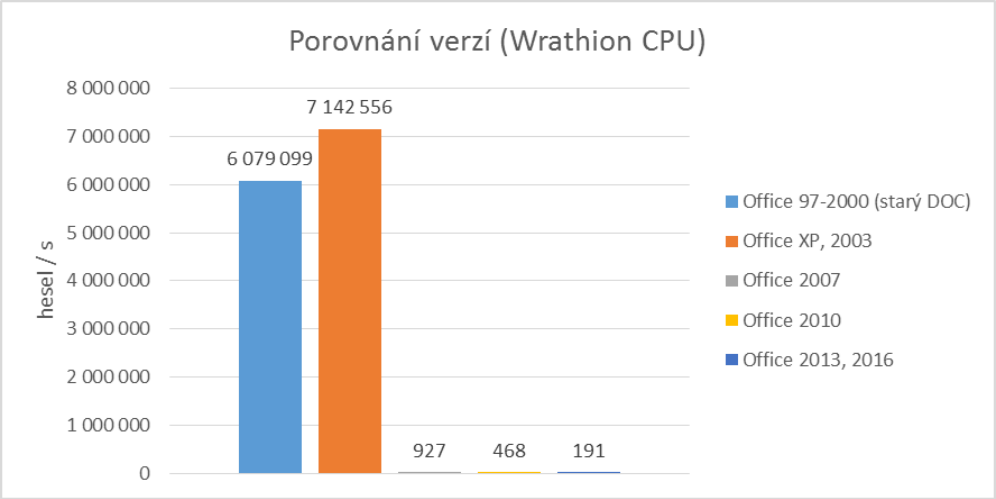
Obrázek C.4: Porovnání CPU a GPU obnovy hesla verze Office 2010



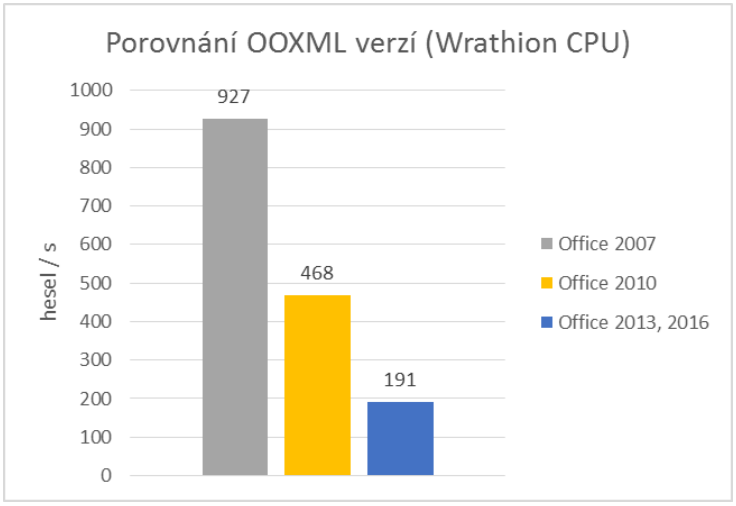
Obrázek C.5: Porovnání CPU a GPU obnovy hesla verze Office 2013 a 2016

Příloha D

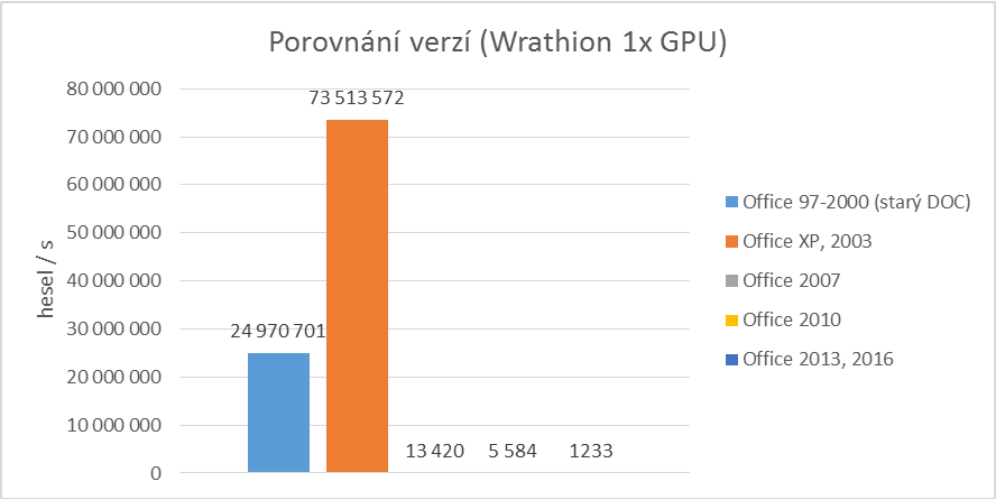
Grafy srovnání verzí Office



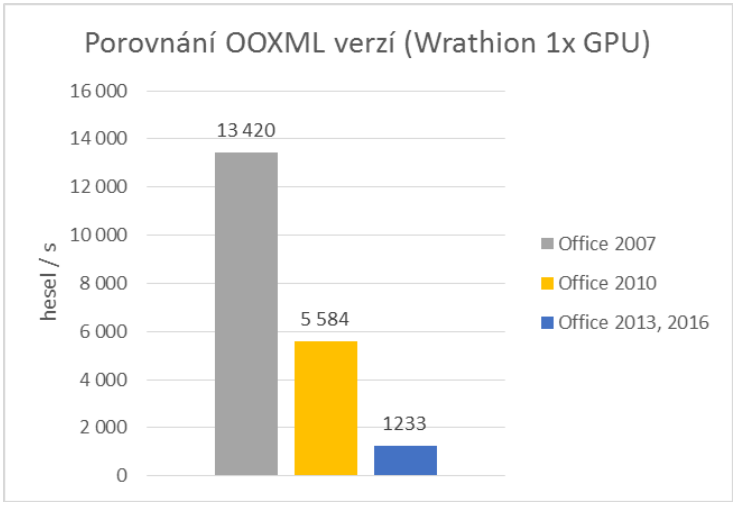
Obrázek D.1: Srovnání jednotlivých verzí Office – CPU



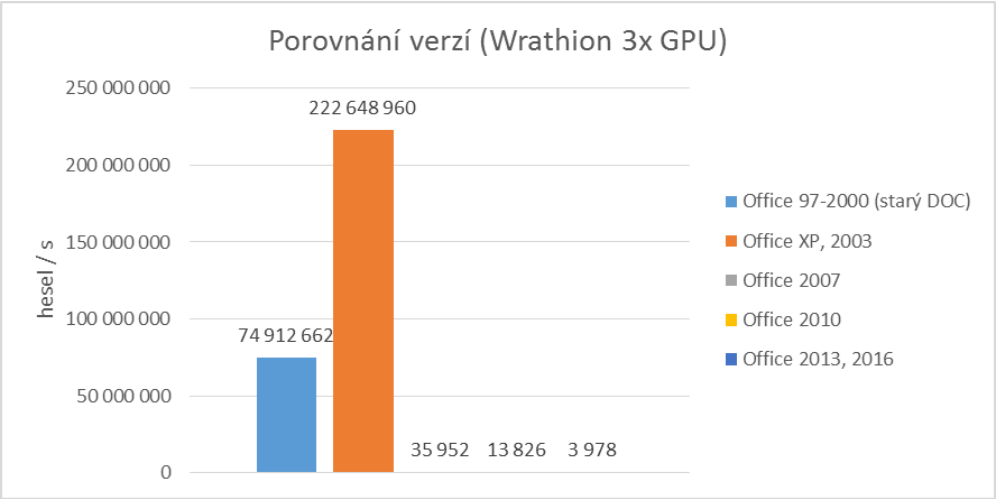
Obrázek D.2: Srovnání jednotlivých OOXML verzí Office – CPU



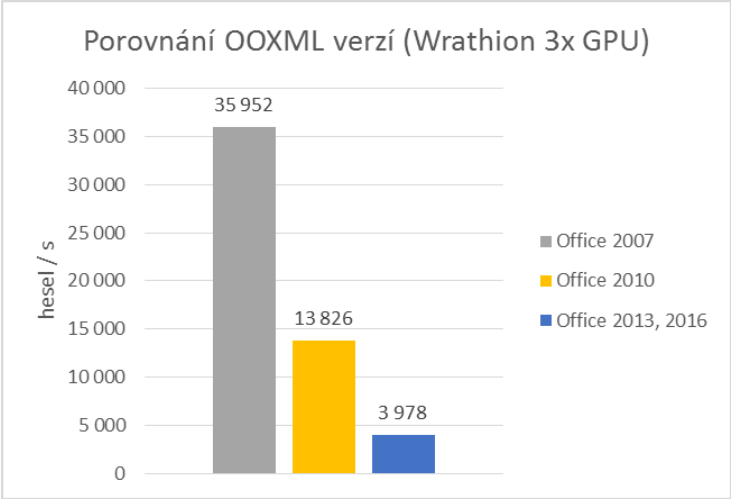
Obrázek D.3: Srovnání jednotlivých verzí Office – 1x GPU



Obrázek D.4: Srovnání jednotlivých OOXML verzí Office – 1x GPU



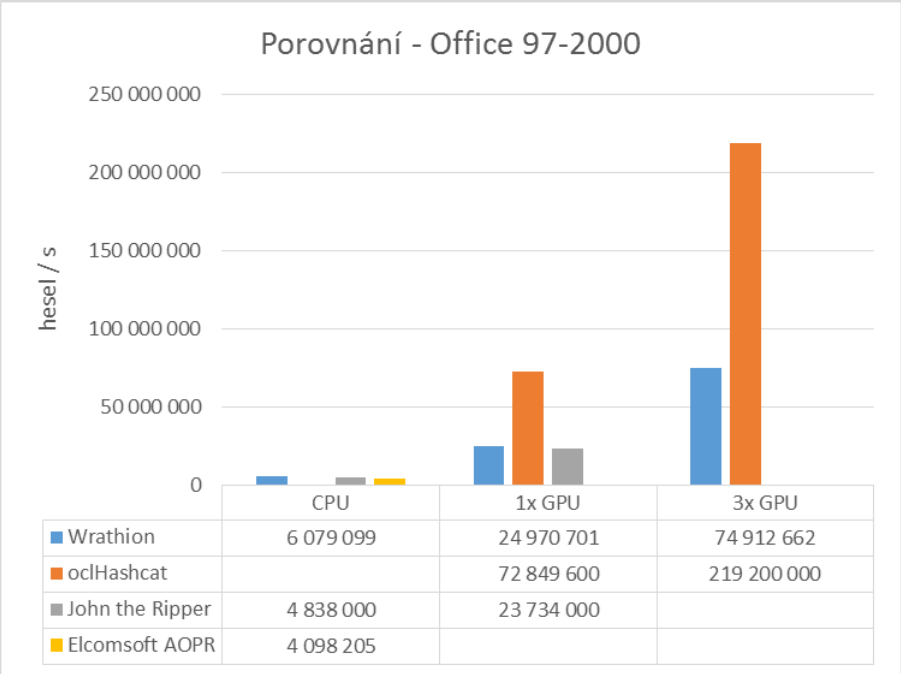
Obrázek D.5: Srovnání jednotlivých verzí Office – 3x GPU



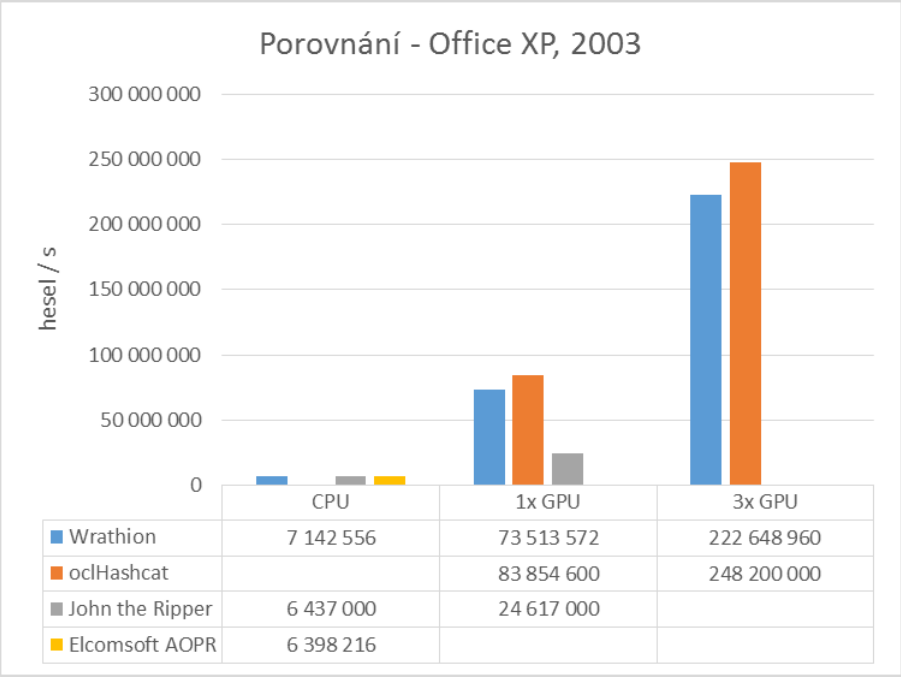
Obrázek D.6: Srovnání jednotlivých OOXML verzí Office – 3x GPU

Příloha E

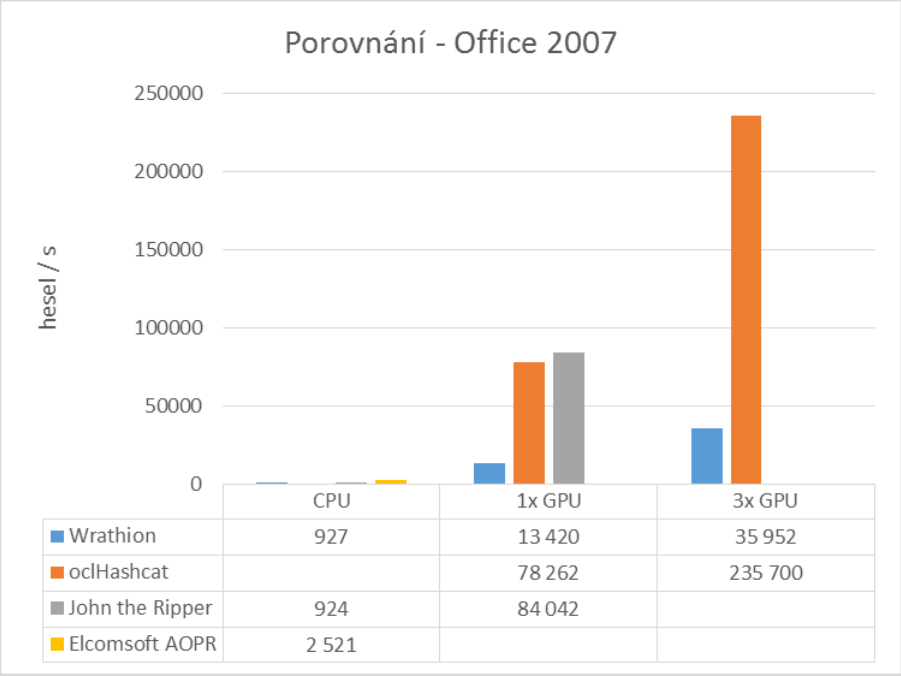
Grafy srovnání s konkurencí



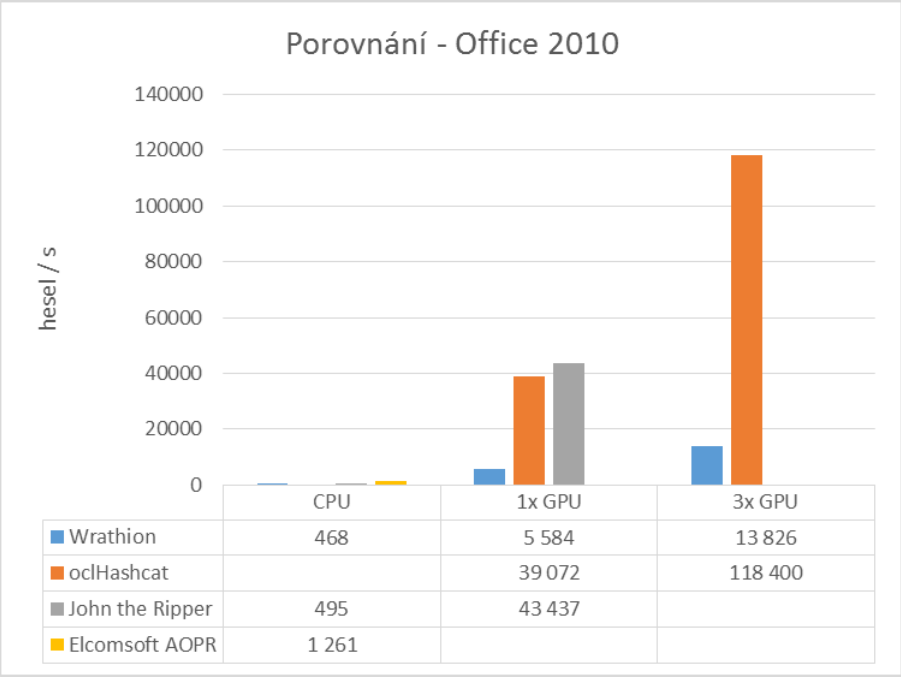
Obrázek E.1: Srovnání verze 97-2000 s konkurenčními nástroji



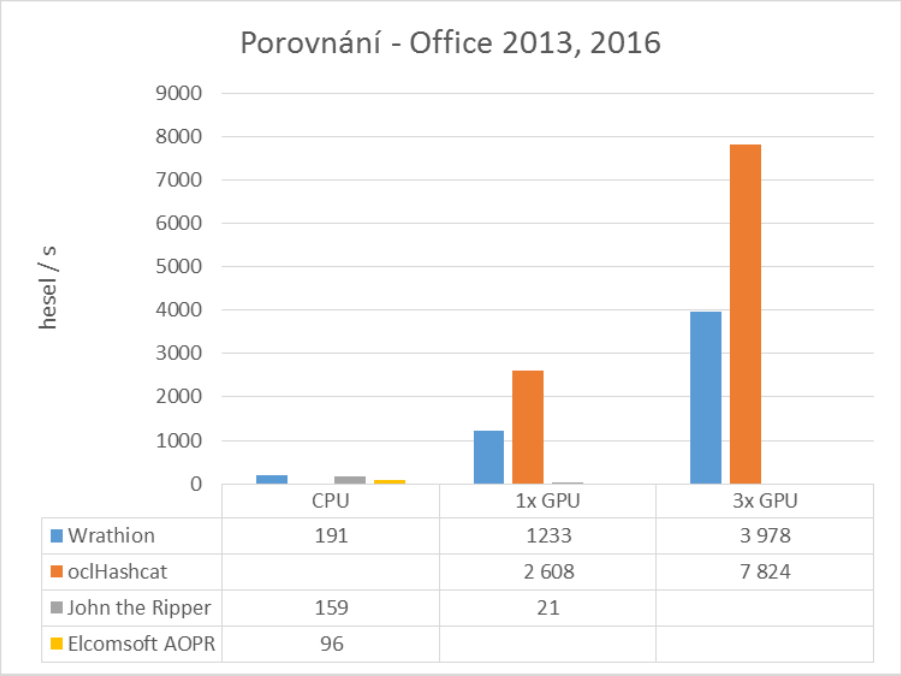
Obrázek E.2: Srovnání verze XP a 2003 s konkurenčními nástroji



Obrázek E.3: Srovnání verze 2007 s konkurenčními nástroji



Obrázek E.4: Srovnání verze 2010 s konkurenčními nástroji



Obrázek E.5: Srovnání verze 2016 s konkurenčními nástroji

Příloha F

Obsah CD

Na přiloženém CD se nachází:

- tato práce v elektronické podobě
- zdrojové kódy této práce v LaTeXu
- zdrojové kódy programu Wrathion společně s návodem na spuštění
- soubor README s informacemi o vytvořených a upravených souborech
- testovací vstupy pro nové moduly nástroje Wrathion