# Experimental Evaluation of Password Recovery in Encrypted Documents

Radek Hranický, Petr Matoušek, Ondřej Ryšavý and Vladimír Veselý
*Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 61266, Brno, Czech Republic*

Keywords: Password Recovery, GPU Acceleration, Privacy, Digital Forensics, Encrypted Documents.

Abstract: Many document formats and archiving tools (PDF, DOC, ZIP) support encryption to protect the privacy of sensitive contents of the documents. The encryption is based on standard cryptographic algorithms as AES, SHA, and RC4. For forensic purposes, investigators are often challenged to analyze these encrypted documents. The task of password recovery can be solved using exhaustive state space search using dictionaries or password generators augmented with heuristic rules to speed up recovery. In our experimental study, we focus on the password recovery of the common document and archiving formats using parallel computation on conventional hardware with multi-core CPUs or accelerated by GPU processors. We show how recovery time can be estimated based on the alphabet, maximal password length and the performance of a given hardware. Our results are demonstrated on Wrathion, a tool developed by our research team.

## 1 INTRODUCTION

File protection is often realized by encrypting the file contents protected by a user-created password. The legitimate use of file protection is to avoid unauthorized access and misuse of the file contents. If the document encryption is a traditional method of protecting and sharing documents with sensitive information, contradicting requirements appear: passwords have to be strong enough in terms of length and a set of chosen characters. On the other hand, it should be easy to be remembered for everyday use. The simplest recovery method is a brute force attack that can eventually find a correct password by generating an exhaustive set of all possible character permutations. Since most of the passwords are human generated, dictionary attacks and rule-based attacks can speed up password recovery significantly.

User habits with respect to password strength were studied mostly in the Internet environment. The results of research by Florencio and Herley presented in (Florêncio and Herley, 2007) give an estimation of the character of passwords utilized for Web sites. The authors considered over a half of million users during three months. The average observed password bit strength was around 40 bits[1]. Users also share their passwords with different sites. In average, a user uses a single password for 5.67 sites. According to another study focused on password habits among American consumers performed by CSID[2], the average password length is about nine characters. In average, 76% of users also reuse their passwords between different sites. Based on this fact, we suppose, plenty of users use their web account passwords to secure documents as well.

How easy it is recover encrypted documents was demonstrated by a study of security measures applied to medical files during clinical trials (Emam et al., 2011). Eman et al. were able to recover 93% of passwords using available commercial tools. In another study (Al-Wehaibi et al., 2011), Al-Wehaibi et al. summarize user behavior towards password usage and conduct a new experiment that coincide with previous observations. According to their data, 65% of passwords are at most eight-character long and approximately 8 out of 10 passwords consist only of alphanumerical characters. Our study reveals how password recovery using brute force attack can be implemented on conventional hardware and with the use of GPU acceleration.

### 1.1 Contribution

This paper evaluates the resilience of password protected documents and archives against password recovery attacks. It shows statistical distribution of

[1] Quantized bit strength is computed as $log_2(|A|^n)$, where $A$ is an alphabet and $n$ is a password length.

[2] See http://www.csid.com/wp-content/uploads/2012/09/CS_PasswordSurvey_FullReport_FINAL.pdf

Hranický, R., Matoušek, P., Ryšavý, O. and Veselý, V.
Experimental Evaluation of Password Recovery in Encrypted Documents.
DOI: 10.5220/0005685802990306
In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, pages 299-306
ISBN: 978-989-758-167-0

characters in available user password datasets and theoretical values of password recovery using massive parallelism. We show experimental results of password recovery on DOC, PDF and ZIP formats. Our approach is to show possible ways of password recovery parallelization and underline the potential of GPU-based computation. Our experiments prove the advantage of a GPU accelerated approach in comparison to a single CPU computation. Experimental results were obtained by our tool Wrathion and other available tools.

### 1.2 Structure of the Paper

The paper is organized as follows. The related research is summarized in section 2. Section 3 discusses the statistical analysis of characters in user passwords and estimation of password recovery time for different password lengths. Section 4 describes the ways of password recovery acceleration using GPUs. Section 5 describes a process of password recovery on a tool we developed with our team. Finally, section 6 brings experimental results of password recovery using different tools and discusses the resilience of documents and archives against brute force recovery techniques.

## 2 RELATED WORK

Since cryptographic algorithms consist of simple integer and binary operations, they can be implemented in the multi-processor environment using OpenCL or CUDA programming as shown in (Marks and Niewiadomska-Szynkiewicz, 2014). Some encryption algorithms can limit parallelism if synchronization is needed as discussed in (An et al., 2015) where a combination of SHA-1 hashing and AES decryption on RAR files is presented. This can be overcome by careful division of password database as shown in (Apostal et al., 2012). For another speed up of password recovery, it is possible to use tables with precomputed hash function tables, e.g., rainbow tables (Thing and Ying, 2003).

There are also advanced automated techniques which consider users to choose easily memorable passwords containing existing words as discussed in (Yampolskiy, 2006). Thus, we can use complex heuristics for generating most-likely passwords, e.g. by using a probabilistic grammar as described in (Weir et al., 2009). Probability-based techniques are used, e.g. by John the Ripper[3], and oclHashcat [4] tools. However, as we have experimentally proven (see section 6), even this approach is not robust enough to deal with randomly-generated passwords.

Non-automated approaches include mathematical analysis, the study of side-channels, and other techniques allowing to exploit the weaknesses of a given algorithm. As shown in (Bergen and Caelli, 1990), on WordPerfect 5.0 documents, the ciphertext-only attack was successfully performed due to the weak design of the document format, which contained enough known plaintext to guess the encryption key in a short period of time.

## 3 STATISTICAL PASSWORD ANALYSIS

The complexity of an attack highly depends on a password length, and alphabet used. An ideal password should have sufficient length and include non-alphanumerical characters. Strongest passwords have a uniform distribution of probability in the occurrence of characters of the source alphabet. However, users tend to create passwords that expose some characteristics in order to ease their memorization. In this section, we present the results of our statistical analysis of passwords used. Such analysis cannot be considered complete since it relies on published datasets of leaked passwords from various services only. For objectivity, we also consider machine generated passwords by *pwgen*[5]. This tool generates secure but easily memorizable passwords.

Statistical analysis of user passwords was created over following datasets[6]: *Myspace* is a password list consisting of 37,000 items obtained from a MySpace phishing attack in 2006; *phpbb* consists of 184,000 passwords stolen from phpbb.com website in 2009; *rockyou* is a list of 14,000,000 of user passwords from RockYou in-game video platform leaked in 2009; *singles* consists of more than 12,000 passwords from singles.org website leaked in 2010; *facebook* comprise of 2,441 passwords stolen by a Facebook phishing attack in 2010; *pwgen1* represents a list of 1,000,000 passwords generated by `pwgen`; and *pwgen2* contains 1,000,000 passwords generated by `pwgen -s` that generates passwords with uniform distribution.

Table 1 shows basic statistics about passwords from different datasets. Interestingly, most of the passwords consists of lower case letters with numerals (lc+nm). Lower case only passwords (lc) build the second most common group regardless of the dataset.

[3] See http://www.openwall.com/john/

[4] See http://hashcat.net/oclhashcat/

[5] See http://sourceforge.net/projects/pwgen/.

[6] See http://wiki.skullsecurity.org/Passwords

Table 1: Statistical analysis of password characters, sizes and lengths.

| | Distribution of character classes in passwords | | | | | | | | | | Dataset statistics | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | nm | lc | lc+nm | uc | uc+nm | bc | bc+sc | bc+nm | bc+nm+sc | other | $\|A\|$ | $\phi$ length | $\phi$ strength |
| myspace | 0.72 | 6.47 | 75.92 | 0.24 | 2.87 | 0.16 | 8.42 | 2.38 | 2.8 | 0.02 | 98 | 8.59 | 41.36 |
| phpbb | 11.24 | 41.24 | 35.70 | 0.93 | 1.19 | 2.68 | 1.02 | 4.82 | 1.16 | 0.02 | 96 | 7.54 | 36.19 |
| rockyou | 16.36 | 25.98 | 42.35 | 1.60 | 2.84 | 1.11 | 3.16 | 2.66 | 3.83 | 0.10 | 711 | 8.75 | 41.36 |
| singles | 8.37 | 55.11 | 26.48 | 2.26 | 0.82 | 4.37 | 0.15 | 2.34 | 0.11 | 0.00 | 63 | 6.74 | 32.90 |
| facebook | 16.35 | 38.47 | 34.29 | 0.66 | 0.53 | 0.70 | 3.24 | 1.52 | 4.1 | 0.16 | 90 | 9.523 | 41.36 |
| pwgen1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 | 0 | 62 | 8 | 47.63 |
| pwgen2 | 0 | 0 | 0 | 0 | 1.6 | 0 | 0 | 98.4 | 0 | 0 | 62 | 8 | 47.53 |

*nm*:numerical only passwords; *lc/uc/bc*:lower/upper/both case only password; *sc*:special characters.

Not so many passwords includes special or non-ascii characters (bc+nm+sp), although, as many as 711 different characters were found in *rockyou* dataset. An average length of passwords does not exceed ten characters. Average bit strength is around 41 what corresponds to (Florêncio and Herley, 2007). Unlike to (Al-Wehaibi et al., 2011), the usage of non-alphanumerical passwords is about 90-97%. This is higher than Al Wehaibi's results. (Mazurek et al., 2013) shows an average length of CMU students' and employee's password to be 10.7 characters with the higher occurence of nm, uc, and sc. We assume university students and staff to be more familiar with security issues than average users of websites denoted above. The CMU research also underlined the significance of positions in password where nm/uc/sc are placed, the more predictable position is chosen, the weaker the password is. (Florêncio et al., 2014) states, that users still frequently use weak passwords like 'password', or 'monkey'. About 1% of RockYou users use '123456' as a password. However, as we mentioned in section 1, 76% of users reuse passwords between different sites, thus we assume there is a high probability that they use the same passwords for document protection.

Table 2 shows how much time it takes to check all possible combinations of alphanumerical passwords ($|A| = 62$) with different lengths. Speed is measured as a number of passwords generated and verified per second. The lowest value (verification of 10,000 passwords per sec) can be easily achieved by a common desktop computer. The highest presented speed (1 billion passwords per sec) can be achieved by medium to large scale distributed computing platforms or supercomputers. The presented values show that for lengths $\geq 9$ the problem becomes intractable. For reduced alphabet, e.g., *lc+nm* class ($|A| = 36$), it is possible to recover passwords up to the length of 10.

Password search time can be reduced if we have additional knowledge of password characteristics. To gain such knowledge, various analyses are possible. Table 3 shows top password characters from analyzed datasets. Each value represents a percentage of the total occurrence in passwords from the given dataset. Although datasets employ different password policies and various kinds of users they have a similar distribution of characters. For example, the letters 'a' and 'i' are the most used pair of letters in all of them, except those generated by pwgen. To refine the analysis, occurrences of letters in different positions in passwords can also provide useful information. From this analysis, it can be seen that some letters have a significantly lower probability of occurrence at certain positions.

The presented statistical evaluation gives information usable for adjusting password recovery methods. Once, we know the approximate characteristics of common passwords the search method can be focused to a subspace of the complete password name-space in order to reduce the password recovery time. It means that combinations of characters that are more probable to occur in a password are tested first. However, this does not work for all passwords. Nevertheless, there is a significant amount of passwords that exhibit these properties and can be found faster.

# 4 GPU ACCELERATION

Fortunately, the password recovery process can be parallelized easily. We can divide the set of posssible passwords to subsets, while the passwords from each subset may be generated on a different computation unit. Also the password verification process could be parallelized. For a single computation unit with the knowledge of the verification value, there is no direct dependency on other passwords generated. This means different passwords can be verified independently of each other. Even some encryption algorithms like AES can be computed parallely (Apostal et al., 2012). This makes the password recovery process an ideal candidate for a parellel and/or distributed environment.

The acceleration method we focus on in this paper is the usage of GPU devices. A GPU is present in every PC, which means no special hardware is required. With a bit of exaggeration, we can state,

Table 2: Speed of exhaustive password search.

| Passwords of alphabet with $\|A\| = 63$ | | | Number of passwords generated and verified per second | | | | | |
|---|---|---|---|---|---|---|---|---|
| length | bit strength | no. of combinations | $10^4$ | $10^5$ | $10^6$ | $10^7$ | $10^8$ | $10^9$ |
| 5 | 29.77 | $916 \cdot 10^6$ | 1 day | 2.5 hours | 15.25 mins | 1.5 mins | 9 secs | 1 sec |
| 6 | 35.72 | $57 \cdot 10^9$ | 66 days | 6.5 days | 16 hours | 1.5 hours | 9.5 mins | 56 secs |
| 7 | 41.68 | $3.5 \cdot 10^{12}$ | 11 years | 1 year | 41 days | 4 days | 10 hours | 58 mins |
| 8 | 47.63 | $218 \cdot 10^{12}$ | 692 years | 69.25 years | 7 years | 253 days | 25.25 days | 60.5 hours |
| 9 | 53.59 | $13.5 \cdot 10^{15}$ | 42 808 years | 4 280 years | 428 years | 42 years | 4.2 years | 156 days |

Table 3: Most used password characters (in %).

| dataset | a | e | i | o | 1 | r | n | s | l | h | 2 | 0 | t | m | 3 | c |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| myspace | 6.70 | 7.18 | 4.57 | 4.83 | 5.79 | 4.28 | 3.94 | 4.19 | 4.07 | 2.54 | 3.11 | 2.43 | 3.39 | 2.69 | 2.15 | 2.82 |
| phpbb | 7.00 | 6.37 | 4.49 | 4.71 | 4.38 | 4.46 | 4.25 | 4.12 | 3.55 | 2.32 | 3.26 | 3.26 | 3.58 | 2.87 | 2.49 | 2.49 |
| rockyou | 7.05 | 5.75 | 4.44 | 4.13 | 5.37 | 3.66 | 3.86 | 3.32 | 3.56 | 1.87 | 4.18 | 4.58 | 2.74 | 2.56 | 3.00 | 2.08 |
| singles | 8.17 | 8.19 | 5.15 | 5.73 | 3.36 | 4.74 | 4.91 | 4.71 | 4.54 | 2.41 | 2.35 | 2.12 | 3.53 | 3.17 | 1.60 | 2.52 |
| facebook | 8.39 | 5.60 | 4.96 | 4.49 | 5.32 | 4.27 | 4.15 | 3.87 | 3.40 | 2.33 | 3.84 | 3.60 | 2.98 | 2.61 | 2.94 | 2.27 |
| pwgen1 | 8.28 | 12.58 | 8.92 | 8.59 | 1.46 | 0.77 | 1.06 | 1.28 | 0.77 | 8.72 | 1.45 | 1.44 | 1.28 | 0.77 | 1.45 | 1.28 |
| pwgen2 | 0.19 | 0.19 | 0.19 | 0.19 | 0.27 | 0.19 | 0.19 | 0.19 | 0.19 | 0.19 | 0.27 | 0.27 | 0.19 | 0.19 | 0.27 | 0.19 |
| all | 7.83 | 7.61 | 5.42 | 5.41 | 4.28 | 3.57 | 3.52 | 3.37 | 3.19 | 3.05 | 3.03 | 2.90 | 2.70 | 2.32 | 2.01 | 1.90 |

that every GPU is, in fact, a little but ”highly-parallel supercomputer”. For example AMD(R) Tri-X R9 290X card contains 2816 stream processors[7]. Using OpenCL (Advanced Micro Devices Inc., 2010) or CUDA (NVIDIA Corporation, 2012) to implement algorithms for GPU, we can perform the password recovery process in a highly-parallel environment.

Independent of AMD or NVIDIA architecture, the computation on GPU is divided into so called *work-items*, while each work-item has its own private memory. A *workgroup* is a collection of work-items with a collective local memory. There is also a global memory on each GPU device which can be accessed from the host machine (Advanced Micro Devices Inc., 2010; NVIDIA Corporation, 2012).

GPU Accelerated computation can be used for both password generation and password verification. It is also possible to perform a hybrid CPU/GPU password recovery process. The scalability potential is high - a single host machine may contain multiple GPU units while multiple host machines may be connected together creating a computation cluster. In such cluster, host machines can communicate by using OpenMPI[8] or another technique.

# 5 PASSWORD RECOVERY

A typical architecture of a password recovery tool as implemented in our tool Wrathion is shown in Figure 1. This architecture includes a core module with password generators and specific modules (crackers) for password recovery of each file format (e.g., DOC, PDF, ZIP). Generation of possible passwords should be independent on recovery file format. Both recovery modules and password generators can support computation on CPU and/or GPU.

The password recovery process is composed of the following steps: (i) document identification and detection of cryptographic algorithm(s), (ii) password generation, and (iii) password verification.

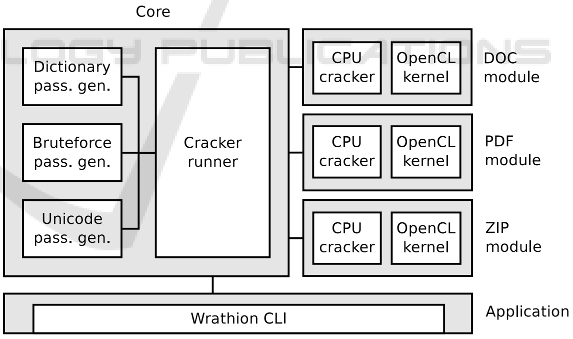

Figure 1: Architecture of Wrathion.

## 5.1 Identification of File Format

Identification of a file format is the first task that has to be performed. Usually, the file format can be identified from within the file by a signature. The signature is a unique string[9] that is mostly specified at the beginning of the file. When the file format identified, it is necessary to parse the document to obtain metadata related to encryption (algorithms used, etc.).

[7] See http://www.sapphiretech.com/

[8] See http://www.open-mpi.org/

[9] See http://www.filesignatures.net/

## 5.2 Password Generation

Password generation is an essential function of any password recovery tool. The generator creates different combinations of strings that are tested (verified) as a functional password. It is necessary for generators to support national alphabets, since some document types may be encrypted with passwords consisting of non-ASCII symbols. Based on observations from previous sections, following types of generators can be employed for password recovery of user documents:

### 5.2.1 Dictionary Password Generator

The dictionary generator takes a given dictionary as a source of possible passwords. Such approach is often successful, since many users use popular passwords like ”123456” or ”qwerty”, as mentioned in (Florêncio et al., 2014). Usage of dictionaries can substantially reduce the password name space.

### 5.2.2 Brute Force Password Generator

In the case of complicated passwords, brute force password generator has to be used. This generator creates every possible permutation of a given alphabet limited by the maximum length.

### 5.2.3 Rule-based Generator

As described in section 2, we can use advanced probability-based techniques to reduce a password name space. This involves letter prediction, digraph/trigraph frequency analysis, letter position guessing, etc. Such generation may be based on a set of heuristics or a probabilistic grammar.

## 5.3 Password Verification

Password verification procedure is usually defined by the document format inventor, and is unique to each document format and its version. In this phase, we use metadata obtained by the first phase (identification). Such metadata usually include a part called *password verification value*. The verification method usually consists of one or more hash algorithms, possibly combined with one or more encryption algorithms. The result is then compared with a verification value. When matched, the password is considered correct. In some cases, an input password has to be combined with a salt, which is usually a particular string located inside the document.

### 5.3.1 Microsoft Office

The encryption technique in Office 95 documents is XOR of the cascaded password with the document content. Based on the fact that only 16-bit key was used, modern password recovery software can find the password instantly.

In Office 97/2000 documents, password verification is performed by using 40-bit RC4 stream cipher in combination with the MD5 hash algorithm. However, when the document gets modified and saved, the initialization vector remains the same, which allows the use of known-plaintext attack, as described in (Wu, 2005).

In Office XP and 2003, the default protection remained the same, but an opportunity to use a custom protection algorithm was added. Starting with Office 2007[10] , AES is used for encryption, and the password verification algorithm consists of 50,000 rounds of SHA-1.

### 5.3.2 Portable Document Format

PDF documents can be secured by two passwords: *user password*, and *owner password*. Owner password is only used to restrict selected operations within the file, and can be ignored. User password is used for encryption of the document.

Like Office 97/2000, for password verification, PDF security revisions from 1 to 4 use the combination of RC4 and MD5. In revision 3, the MD5 is performed 50 times, as described in (Adobe Systems Incorporated, 2008b).

In security revision 5, password in UTF-8 encoding is combined with salt and processed with a single round of SHA-256 as described in (Adobe Systems Incorporated, 2008a). As we have proven in section 6, this approach is weaker than in revision 4.

Security revision 6 (PDF 2.0) uses strong password verification algorithm described by ISO 32000-2, but this specification is not yet publicly available. However, a pseudocode of the algorithm has leaked[11].

### 5.3.3 ZIP

The original Zip 2.0 (PKZIP) uses a very simple cipher, which is also vulnerable to known-plaintext attack. Password verification is done by comparing the 8-bit value obtained in the initialization phase of the

[10] See https://msdn.microsoft.com/en-us/library/cc313105.aspx

[11] See http://esec-lab.sogeti.com/posts/2011/09/14/the-undocumented-password-validation-algorithm-of-adobe-reader-x.html

algorithm with the lowest 8 bits of file CRC checksum.

Modern ZIP formats[12] use AES encryption mostly with 128/192/256 bit key. Password verification is performed by using PBKDF2 and HMAC-SHA1 algorithms.

# 6 PRELIMINARY RESULTS

In this section, we present experimental results of Wrathion. For each module, we performed a measurement of performance and time, and calculated the achieved GPU acceleration in comparison with the CPU-only method. The testing machine contained Intel(R) Core i7 CPU 920 @ 2.67Ghz processor, 16 GB of DDR3 RAM, and two AMD Tri-X R9 290x GPU cards.

We also compared Wrathion with other password recovery tools: oclHashcat, John the Ripper 1.8.0 - jumbo 1 (John), Elcomsoft[13] Advanced Office Password Recovery 6.10 (AOPR), Elcomsoft Advanced PDF Password Recovery 5.06 (APPR), and Elcomsoft Advanced Archive Password Recovery 4.54 (AAPR). Since AAPR has only CUDA-based acceleration, for this case, we used NVIDIA GeForce GTX 660Ti instead. Since oclHashcat is GPU-only, and its predecessor Hashcat does not support any of the formats analyzed, we could not determine the acceleration of GPU.

## 6.1 Performance of Wrathion

First three lines of Table 4 show the performance (*perf.*) in passwords per second, and GPU acceleration (*acc.*) of Wrathion. The highest acceleration was achieved on ZIP archives encrypted with AES. A single-GPU recovery was 30.28 times faster than using CPU method. With dual-GPU deployment, the recovery was 60.56 times faster than with CPU. ZIP-AES uses PBKDF2 for key generation in combination with SHA-1 hash function. These algorithms offer plenty of space for parallelization, which was utilized efficiently in our OpenCL kernels. There is almost no time difference between the password recovery of ZIP encrypted with AES-128 and AES-256 since AES is only the encryption algorithm, but for password verification, PBKDF2 and SHA-1 algorithms are used.

DOC and PDF revisions 3 and 4 use RC4 cipher with MD5 hash function repeated multiple times. Since RC4 has high memory requirements, we made our kernels faster by putting the whole S-BOX inside a GPU local memory. Another 12% of speedup was achieved by storing the key in a vector inside a standard array.

In PDF revisions 3 and 4, the MD5 hash is computed from the array of values for key alignment concatenated with an ID from the document footer. This gives Wrathion the opportunity to calculate the hash on CPU before the password recovery process is started and then send it to the GPU since this value does not change during the recovery process.

## 6.2 Performance Comparison

Table 4 also shows the performance and GPU acceleration of other tools. For each of the document formats we selected one encryption method for chart illustration.

Figure 2 shows the performance of oclHashcat, AOPR, John, and Wrathion. The document type chosen was MS Office DOC 97/2000. Within this old document format, AOPR and John evinced rather a poor performance. OclHashcat seemed to be much more efficient than AOPR, but is behind Wrathion, possibly because its creators considered DOC format obsolete and have not optimized their tool much for it.

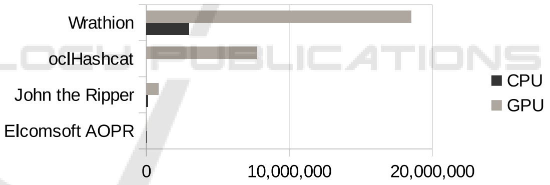

Figure 2: Performance of tools (DOC 97/2000).

Figure 3 shows the performance of AOPR, oclHashcat, and Wrathion on PDF 1.7 revision 4. Within CPU password recovery, the performance difference between AOPR and Wrathon was not excessively radical. However, with GPU acceleration, Wrathion and oclHashcat turned out to be much more efficient. For revision 4, Wrathion and oclHashcat have similar performance. However, for revision 5 (See Table 4.), oclHashcat turned out to be faster than Wrathion.
Figure 4 compares AAPR, John, and Wrathion on ZIP encrypted by AES-256. Unfortunately, oclHash-

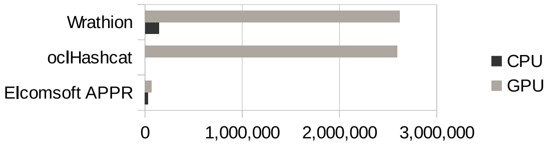

Figure 3: Performance of tools (PDF R4).

[12] See https://pkware.cachefly.net/webdocs/APPNOTE/APPNOTE-6.3.3.TXT

[13] See http://www.elcomsoft.com/

Table 4: Performance (pass./s) and GPU acceleration using different tools on document formats.

| | ZIP AES-128 | | ZIP AES-256 | | DOC 97/2000 | | PDF Rev 4 | | PDF Rev 5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Tool | perf. | acc. | perf. | acc. | perf. | acc. | perf. | acc. | perf. | acc. |
| Wrathion (CPU) | 4.3k | - | 4,329 | - | 2,985k | - | 143k | - | 7,425k | - |
| Wrathion (1x GPU) | 131k | 30.34 | 131k | 30.28 | 18,548k | 6.21 | 2,622k | 18.36 | 82,735k | 11.14 |
| Wrathion (2x GPU) | 262k | 60.69 | 262k | 60.56 | 35,262k | 11.81 | 4,522k | 31.67 | 136,913k | 18.44 |
| oclHashcat (1x GPU) | n/a | - | n/a | - | 7,766k | - | 2,596k | - | 144,074k | - |
| oclHashcat (2x GPU) | n/a | - | n/a | - | 15,377k | - | 5,121k | - | 286,340k | - |
| John (CPU) | 649 | - | 376 | - | 115k | - | n/a | - | n/a | - |
| John (1x GPU) | 28.7k | 44.28 | 14.8k | 39.38 | 861k | 7.5 | n/a | - | n/a | - |
| Elcomsoft (CPU) | 8.1k | - | 8.1k | - | 20k | - | 33k | - | 28,861k | - |
| Elcomsoft (1x GPU) | n/a | - | n/a | - | 26k | 1.31 | 69k | 2.1 | n/a | - |

cat does not support ZIP format, and AAPR is CPU-only. However, for CPU, AAPR showed the best performance within all tools tested.

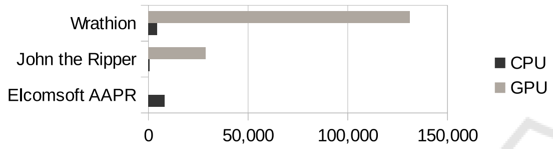

Figure 4: Performance of tools (ZIP AES-256).

## 6.3 Time Comparison

We also measured the real recovery times of Wrathion and other tools. In this experiment, all tools were in brute-force mode with the lowercase Latin alphabet (a-z). The results for lengths from 4 to 7 characters (from 'acz' to 'aczqfbi') are denoted in table 5. It is worth to compare these results with theoretical estimations from Table 2. The recovery time of shorter passwords oscillated around 1 second while recovery of longer passwords would take even days. In cells marked with 'app.' the time was approximated due to a long duration. Cells marked with 'err' show cases where we were not able to succesfully recover the correct password (after many attempts) due to a possible bug in the program leading to an error.

The results shown in Table 5 relatively correspond to previously measured performances. Wrathion needs at least 1 seconds since it is the interval for communication with working threads. For encrypted DOC, AOPR has shown an anomaly when generating a password of 6 and more characters. Even though we defined the alphabet as lowercase Latin letters, for some reason, the passwords were generated from numbers, etc. As mentioned above, AAPR supports only CUDA-based acceleration. However, even with NVIDIA card with CUDA support, the recovery of PDF Revision 5 always ended with a program error.

We also encountered another anomaly connected with ZIP-AES. Using Wrathion and Elcomsoft tools, recovery times of 128-bit and 256-bit AES were fairly comparable (due to the password verification method used). However, with John, the recovery of 256-bit AES took about a double time period than with 128-bit version of AES. We do not yet know the cause of this behavior.

Table 5: Measurement of password recovery time using different tools.

| DOC 97/2000 | | | | |
|---|---|---|---|---|
| tool | 4 | 5 | 6 | 7 |
| Wrathion (CPU) | 1.00s | 1.00s | 5.02s | 1m 52s |
| Wrathion (GPU) | 1.00s | 1.00s | 1.00s | 17.00s |
| oclHashcat (GPU) | < 1s | < 1s | 2.36s | 40.6s |
| John (CPU) | 0.87s | 5.07s | 1m 52s | 40m 50s |
| John (GPU) | 3.06s | 3.61s | 14.92s | 4m 4s |
| Elcomsoft (CPU) | 6.12s | 25.41s | err | err |
| Elcomsoft (GPU) | 5.40s | 19.35s | err | err |
| ZIP AES 128-bit | | | | |
| tool | 4 | 5 | 6 | 7 |
| Wrathion (CPU) | 5.06s | 1m 55s | 49m 40s | 21h 29m |
| Wrathion (GPU) | 2.01s | 5.27s | 1m 40s | 42m 35s |
| John (CPU) | 29.28s | 11m 27s | 4h 47m | app. 6d |
| John (GPU) | 6.05s | 22.23s | 7m 27s | 3h 12m |
| Elcomsoft (CPU) | 2.33s | 59.77s | 26m 24s | 10h 23m |
| ZIP AES 256-bit | | | | |
| tool | 4 | 5 | 6 | 7 |
| Wrathion (CPU) | 5.03s | 1m 55s | 49m 41s | 21h 29m |
| Wrathion (GPU) | 2.01s | 6.24s | 1m 41s | 43m 22s |
| John (CPU) | 54.76s | 21m 54s | 9h 29m | app. 10d |
| John (GPU) | 6.75s | 38.01s | 14m 28s | 6h 17m |
| Elcomsoft (CPU) | 2.30s | 1m 1s | 26m 5s | 10h 27m |
| PDF Revision 4 | | | | |
| tool | 4 | 5 | 6 | 7 |
| Wrathion (CPU) | 1.03s | 4.03 s | 1m 30s | 38m 50s |
| Wrathion (GPU) | 1.00s | 1.00s | 6.00s | 2m 10s |
| oclHashcat (GPU) | < 1s | < 1s | 26.68s | 2m 11s |
| Elcomsoft (CPU) | 0.42s | 7.66 s | 3m 17s | 2h 49m |
| Elcomsoft (GPU) | 0.49s | 7.12 s | 3m 15s | 1h 23m |
| PDF Revision 5 | | | | |
| tool | 4 | 5 | 6 | 7 |
| Wrathion (CPU) | 1.00s | 1.00s | 2.00s | 45.04s |
| Wrathion (GPU) | 1.00s | 1.04s | 1.03s | 4.04s |
| oclHashcat (GPU) | <1s | <1s | < 1s | 2.32s |
| Elcomsoft (CPU) | 0.67s | 0.62s | 0.95s | 12.37s |
| Elcomsoft (GPU) | err | err | err | err |

## 7 CONCLUSION

The basic method of password recovery is an exhaustive search. Despite a huge number of all permutations, currently available hardware and the strength of common passwords make this method still relevant.

With the support of high-end GPUs, the password recovery process can be performed in a fraction of time in comparison with CPU-only computation. In the area of password recovery, there is also a lot of space for parallelization and the scalability potential of the process is high.

In the future, we want to focus mainly on password recovery in distributed environment. We also plan to extend Wrathion with modules for another file formats, and with more sophisticated password generators. Finally, we want to compare our tool with another software, e.g. AccessData Password Recovery Toolkit[14].

## ACKNOWLEDGEMENTS

Research presented in this paper is supported by project "Modern Tools for Detection and Mitigation of Cyber Criminality on the New Generation Internet", no. VG20102015022 granted by Ministry of the Interior of the Czech Republic and a project "Research and application of advanced methods in ICT", no. FIT-S-14-2299 granted by Brno University of Technology.

## REFERENCES

Adobe Systems Incorporated (2008a). *Adobe Supplement to the ISO 32000, BaseVersion: 1.7, ExtensionLevel: 3*. Adobe Systems Incorporated.

Adobe Systems Incorporated (2008b). *Document management Portable document format Part 1: PDF 1.7*. Number 32000-1:2008. International Organization for Standardization, Geneva, Switzerland.

Advanced Micro Devices Inc. (2010). *Introduction to OpenCL programming - Training guide*. Number 137-41768-10. Advanced Micro Devices Inc.

Al-Wehaibi, K., Storer, T., and Glisson, W. B. (2011). Augmenting password recovery with online profiling. *Digit. Investig*, 8:S25–S33.

An, X., Zhao, H., Ding, L., Fan, Z., and Wang, H. (2015). Optimized password recovery for encrypted RAR on GPUs. *CoRR*.

Apostal, D., Foerster, K., Chatterjee, A., and Desell, T. (2012). Password recovery using MPI and CUDA. In *Proc. of HiPS 2012*, pages 1–9.

Bergen, H. A. and Caelli, W. J. (1990). File Security in WordPerfect 5.0.

Emam, K. E., Moreau, K., and Jonker, E. (2011). How strong are passwords used to protect personal health information in clinical trials? *Journal of Medical Internet Research*, 13(1).

Florêncio, D. and Herley, C. (2007). A large-scale study of web password habits. In *Proc. of the 16th Int. Conference on WWW*, pages 657–666.

Florêncio, D., Herley, C., and Oorschot, P. C. V. (2014). An Administrator's Guide to Internet Password Research. In *Proceedings of the 28th USENIX Conference on Large Installation System Administration*, LISA'14, pages 35–52. USENIX Association.

Marks, M. and Niewiadomska-Szynkiewicz, E. (2014). Hybrid cpu/gpu platform for high performance computing. In *Proc. of the 28th ECMS*, pages 523–537.

Mazurek, M. L., Komanduri, S., Vidas, T., Bauer, L., Christin, N., Cranor, L. F., Kelley, P. G., Shay, R., and Ur, B. (2013). Measuring Password Guessability for an Entire University. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 173–186. ACM.

NVIDIA Corporation (2012). *NVIDIA CUDA C Programming Guide*. NVIDIA Corporation.

Thing, V. L. and Ying, H.-M. (2003). Making a faster cryptanalytic time-memory trade-off. *Advances in Cryptology*, pages 617–630.

Weir, M., Aggarwal, S., de Medeiros, B., and Glodek, B. (2009). Password Cracking Using Probabilistic Context-Free Grammars. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 391–405.

Wu, H. (2005). The misuse of rc4 in microsoft word and excel.

Yampolskiy, R. (2006). Analyzing user password selection behavior for reduction of password space. In *Carnahan Conferences Security Technology, Proceedings 2006 40th Annual IEEE International*, pages 109–115.

[14] http://accessdata.com