

**CERIAS Tech Report 2015-01**  
**The Weakness of WinRAR Encrypted Archives to Compression Side-channel Attacks**  
by Kristine Arthur-Durett  
Center for Education and Research  
Information Assurance and Security  
Purdue University, West Lafayette, IN 47907-2086

THE WEAKNESS OF WINRAR ENCRYPTED ARCHIVES  
TO COMPRESSION SIDE-CHANNEL ATTACKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Kristine Arthur-Durett

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2014

Purdue University

West Lafayette, Indiana

I would like to dedicate this work to my husband, James, and our children Maeke'a and Henry for their love, patience and support.

## ACKNOWLEDGMENTS

I would first like to express my gratitude to the members of my committee for providing me with guidance throughout the process of developing my thesis. In particular, I would like to thank Dr. Eugene Spafford for introducing me to the problem within and providing me with resources to begin my research. I extend warm thanks to Dr. Melissa Dark for introducing me to the field of Information Security and providing support and advice throughout my time in the program. I would like to thank Dr. Samuel Wagstaff for his insightful recommendations as well as providing me with the foundational knowledge that I needed. I would also like to extend acknowledgement of Special Agent Michael Alford's contributions in providing information regarding practical issues and current methods in the problem space.

## TABLE OF CONTENTS

	Page
LIST OF TABLES . . . . .	vi
LIST OF FIGURES . . . . .	vii
ABSTRACT . . . . .	viii
1 INTRODUCTION . . . . .	1
1.1 Related Work . . . . .	2
2 WINRAR . . . . .	6
2.1 WinRAR v5.0 . . . . .	6
2.2 WinRAR encryption . . . . .	7
2.3 WinRAR compression . . . . .	8
2.3.1 LZSS . . . . .	8
2.3.2 PPMII . . . . .	9
2.3.3 Intel IA-32 . . . . .	9
2.3.4 Delta encoding . . . . .	10
3 METHODS . . . . .	11
3.1 Compression ratios . . . . .	11
3.2 File detection . . . . .	12
3.3 Man-in-the-Middle attack . . . . .	15
3.3.1 RAR5 file header . . . . .	17
4 RESULTS . . . . .	19
4.1 Compression ratios . . . . .	19
4.2 File detection . . . . .	21
4.2.1 Appearance of substrings . . . . .	22
4.2.2 Difference of ratios . . . . .	23
4.2.3 Man-in-the-Middle . . . . .	24

	Page
5 SUMMARY . . . . .	28
5.0.4 Discussion . . . . .	28
5.0.5 Countermeasures . . . . .	31
5.0.6 Conclusion and open questions . . . . .	31
REFERENCES . . . . .	33
A Compression Corpa . . . . .	36
B RAR file header . . . . .	38

## LIST OF TABLES

Table	Page
3.1 Sample of compression ratio data. . . . .	13
3.2 Number of repetitions of text strings of indicated length. . . . .	14
4.1 Descriptive statistics for compression ratio data . . . . .	20
4.2 ANOVA table for comparing compression ratios of different file types .	20
4.3 Tukey's comparison of treatment means . . . . .	21
4.4 95% Confidence Intervals for different file type compression ratios . . .	21
4.5 SAS output of correlation between size and appearance of substrings where the file is present . . . . .	22
4.6 SAS output of correlation between size and appearance of substrings where the file is not present . . . . .	22
4.7 Hypothesis testing results for different levels of $\alpha$ . . . . .	24
A.1 Details of compression testing files . . . . .	36
B.1 RAR file header fields . . . . .	38

## LIST OF FIGURES

Figure	Page
3.1 A RAR5 archive with packed size and compression information highlighted	18
4.1 Box Plot of the distributions of different file types. . . . .	19
4.2 The original <i>alice29.rar</i> archive with the compression method circled and the total file size inside the rectangle . . . . .	24
4.3 The modified <i>alice29-prime.rar</i> with the compression method circled and the total file size inside the rectangle . . . . .	25



## ABSTRACT

Arthur-Durett, Kristine MS, Purdue University, December 2014. The weakness of WinRAR encrypted archives to compression side-channel attacks. Major Professor: Eugene Spafford.

This paper explores the security of WinRAR encrypted archives. Previous works concerning potential attacks against encrypted archives are studied and evaluated for practical implementation. These attacks include passive actions examining the effects of compression ratios of archives and the files contained, the study of temporary artifacts and active man-in-the-middle attacks on communication between individuals. An extensive overview of the WinRAR software and the functions implemented within it is presented to aid in understanding the intricacies of attacks against archives.

Several attacks are chosen from the literature to execute on WinRAR v5.10. Select file types are identified through the examination of compression ratios. The appearance of a file in an archive is determined through both the appearance of substrings in the known area of an archive and the comparison of compression ratios.

Finally, the author outlines a revised version of an attack that takes advantage of the independence between the compression and encryption algorithms. While a previous version of this attack only succeeded in removing the encryption from an archive, the revised version is capable of fully recovering an original document from a encrypted compressed archive. The advantages and shortcomings of these attacks are discussed and some countermeasures are briefly mentioned.

## 1. INTRODUCTION

Malware droppers are Trojans used as container files to deliver files onto a destination host computer [1]. In the field of digital forensics, a dropper may be implemented to obscure data relevant to a crime. Additionally, individuals can use compressed archives in corporate espionage cases where large amounts of data is removed from a system. Compression software such as WinZip and WinRAR are popular choices for concealing incriminating information. Both software packages offer encryption in addition to data compression, which makes them ideal for these purposes.

The use of compression and encryption creates an issue for forensic investigators who may need to access archived files for valuable information. Password search attacks and dictionary attacks are commonly used methods to gain access to an archived file. With some software packages, such as WinZip versions prior to 9.0, the encryption function is weak to these attacks [2]. However, for passwords with length longer than six characters, WinRAR appears secure [3]. Attacks against the encryption itself, such as related-key attacks introduced by Biryukov et al, exist [4,5].

In an effort to provide knowledge about an archive's content to investigators, this paper will explore alternative attacks against the WinRAR software. These include examination of side-channels and exploitation of the interaction between the compression and encryption functions. While recovering the full contents of an archive may not be possible with these attacks, the intention is to reveal information about the contents. This may provide the knowledge that an investigator needs or assist in determining whether password cracking efforts are worthwhile on an archive.

## 1.1 Related Work

Attacks against the encryption of an archive are a natural starting point to consider. The goal of attacks against the Advanced Encryption Standard (AES) is to recover the key used in the algorithm. The key can then be used to decrypt the contents of an encrypted file or message. There are a variety of methods, such as Meet-in-the-Middle, differential or related-key attacks, that have been introduced to discover the secret key.

Meet-in-the-Middle attacks require pairs of plaintexts and their corresponding ciphertexts. The attacker will attempt to decrypt the ciphertext while simultaneously encrypting the plaintext with the hope of finding a key that will cause these operations to converge. Demirci and Selçuk provide an outline to a Meet-in-the-Middle attack on 8-round AES-256 [6]. This attack is shown to have complexity of  $2^{200}$ .

Another class of attacks are a form of differential analysis called impossible differentials. In contrast to the original differential attacks which look for characteristics that hold true with a high probability, impossible differentials look for extremely low-probability differentials. Once identified, these characteristics can be used to recover the key. Lu and Dunkelman introduce an impossible differential attacks that is effective on 8-round AES with a complexity of  $2^{229.7}$  [7].

Finally, Biryukov introduces several variations of related-key attacks against AES-256 with considerable improvements in time complexity [4, 5, 8]. In the related-key model, the attacker uses several keys with a known relation between them. When these keys are used in the encryption function, the attacker is able to trace characteristics of the function induced by the relationship. From this, the key can be recovered. Biryukov et al present a practical attack that is capable of recovering the key for a 9-round version of AES-256 in only  $2^{39}$  time [4]. Biryukov also presents a related-key attack that works in conjunction with a boomerang attack to recover the key from full 14-round AES-256 in  $2^{99.5}$  time [5].

It is important to note that none of the attacks outlined above provide a practical method for attacking WinRAR archives. The majority of the attacks do not work on the full 14 rounds of AES-256, which limits their usefulness. The time complexities are also an issue. The majority of the attacks are simply too computationally expensive to implement.

Beyond attacks on the key space of a WinRAR archive, there are alternative methods to gain information about the contents of an archive or the activities of the owner. These include exploitation of the independence between compression and encryption, the examination of compression ratios and artifacts in temporary folders. Each potential attack is discussed in detail below.

In their paper, Yeo and Phan discuss several attacks based on previous work by Kohno [9, 10]. The first attack involving manipulating the interaction between the compression and encryption algorithm is of particular interest. The attack is as follows. Two individuals, Alice and Bob, share an encrypted compressed archive. A malicious individual, Eve, intercepts the archive in transit and modifies the indicated compression method in the RAR archive's file header. When Bob attempts to decrypt and decompress the modified archive using his secret password, he obtains a compressed version of the original file. The compressed version looks like a corrupted file to Bob, who was expecting to obtain the plaintext of the original file after using his password. Bob then sends the decrypted compressed file he obtained back to Alice to discover the source of the confusion. Eve intercepts once again to obtain the decrypted compressed file, which can be used to reconstruct the original.

This attack relies on the ability of the adversary to intercept communications between Alice and Bob to obtain the required files. However, it is not uncommon for individuals to email files back and forth with little regard to eavesdropping. Therefore it is sufficient to show that this attack holds with the assumption that the necessary files are acquired through other means. Yeo and Phan have verified this attack on WinRAR v3.42 and v2.9. One issue is the fact that in v3.42 only half of the file contents are recoverable due to verification on the length of the file. The effectiveness

of this attack on later versions of WinRAR as well as the newest file format remains to be seen.

File compression provides side-channels that leaks information about an archive's contents, even when encryption is applied. Polimirova-Nikolova showed that the initial size and extension of an archived object relates to the size of the archive itself [11]. Kelsey explores various attacks via the compression side-channel to leak information about the plaintext within an archive [12]. These findings imply that through the passive observation of compression ratios, it is possible to identify file types within the archive. Compression ratios can be viewed through two methods in WinRAR archives. The Info button in the WinRAR graphical interface can provide information on the overall compression ratio for an archive. Further details, including information for individual files contained within an archive, can be found by inspecting the file header. Further research into the ratios that WinRAR yields as well as the effect of multiple file types within an archive is needed to evaluate the effectiveness of this attack.

Less passive attacks allows for the possibility of string detection. Given a set of encrypted compressed messages, it is possible to determine whether an uncompressed plaintext string,  $S$ , appears in the set. This attack requires the encrypted compressed versions of  $S$  appended to the original messages. It may not be feasible to obtain these messages. However, an alternative attack involving the correlation between appearances of substrings of  $S$  within a known file from an archive may be feasible [12].

Finally, examination of a computer's memory is another method discussed in the literature. This can yield information about the archive and its contents. Both Ji-Zhong and Maartmann-Moe note that cryptographic keys may be found in virtual memory [13,14]. There is also evidence that WinRAR stores information in areas such as the windows registry, log files, or temp files [15,16]. A difficulty with identifying cryptographic keys in this method is the fact that in session-based encryption such as WinRAR, the keys are short-lived. When the session is closed, the key is wiped from memory. Maartmann-Moe's experiments were unable to retrieve information

on cryptographic keys from memory. The short time window that this attack must take place in presents further difficulties and leaves this attack impractical for most implementations.

WinRAR leaves behind artifacts that provide information on the user's activities in the archive. Fellows showed that v3.x releases of WinRAR leave artifacts in temp folders that show changes to the archive and files that the user viewed through WinRAR [15]. While exploring the collection of artifacts, Gupta and Mehtre also found that with normal use, information can be found in windows registry, the AppData folder, and Temporary folders. However, this can be avoided by the use of a portable version of the software [16].

## 2. WINRAR

WinRAR is capable of supporting all popular compression formats, including .rar and .zip files [17]. The software uses the Advanced Encryption Standard (AES) to encrypt archives. WinRAR 5.0 and higher supports AES-256 while earlier versions use AES-128. Users specify a password to encrypt the archive in question. The AES key is then derived from the given password implementing Password-Based Key Derivation Function 2 (PBKDF2) [18].

The compression and encryption functions in WinRAR are independent of each other. Files are first compressed then encrypted when added to an archive [9]. The user may further specify one of two encryption modes to apply to the archive. First, the user may encrypt only the file data. This allows information such as file names to be viewed in plaintext. The second mode encrypts both file data and header information, including file names, sizes, and other attributes [19].

### 2.1 WinRAR v5.0

As of September 2013, WinRAR introduced the new RAR5 archiving format. Several important changes, which will be discussed below, are implemented in the newest version. It is important to note that RAR5 files are not compatible with versions of WinRAR prior to 5.0. During this transition to a new format, the older RAR format is currently the default archive format.

Versions 5.0 and above introduce new features to the compression algorithm [18]. The maximum dictionary size has been increased to 1GB. The default size is now 32MB. This gives a higher compression ratio with a sacrifice to speed when compared to the earlier versions. In addition to the general compression algorithms, Intel IA-32 executable and delta compression algorithms are now implemented. Some older algo-

rithms such as RAR 4.x test, audio, true color and itanium are no longer supported. These changes increase the efficiency of the software when handling modern data types.

Changes in the encryption algorithm and related features offers stronger information security. The encryption algorithm now uses 256-bit AES in place of the previous 128-bit AES. To derive the key for AES-256, WinRAR now implements the key derivation function PBKDF2 using HMAC-SHA1. To circumvent the discovery of encrypted information through system memory, the password verification method now allows for the detection of wrong passwords without unpacking the encrypted file. Additionally, the file checksums are now modified with a propriety password-dependent algorithm. According to Rarlabs, it is now “impossible” to guess the contents of a file by comparing it with the typical CRC32 and BLAKE2 values [18]. Users can use a 256-bit length BLAKE2 hash in lieu of the default CRC32 file checksum.

Finally, the RAR5 format has improved the recovery of broken archives. The new implementation is now based on Reed-Solomon error correction codes [18]. If the recovery record is at least 5% of the original file size, the correction scheme provides much higher resistance. This allows the software to detect larger deletions and insertions to an archive. Further details can be found in [20].

## 2.2 WinRAR encryption

WinRAR has used AES encryption beginning with the release of version 3.00 [21]. AES was introduced by the National Institute of Standards and Technology (NIST) in 2001 [22]. It is a symmetric block cipher based on the Rijdael cipher developed by Joan Daemen and Vincent Rijmen.

AES-128 uses a 128-bit length key and consists of 10 rounds while AES-256 has a key length of 256 bits and goes through 14 rounds. Each AES round consists of four transformations [22]:

**SubBytes** This is a non-linear byte substitution using a substitution table.



**ShiftRows** The final three rows of the state are cyclically left shifted.

**MixColumns** Each column is multiplied modulo  $x^4 + 1$  to mix the bytes.

**AddRoundKey** The final transformation is an XOR of the state with the round key.

The final round omits the MixColumns transformation.

The AES algorithm requires a cryptographic key. In a password protected archive, a key can be generated from the password using a key derivation function (KDF). KDFs take as input the password, salt, and the desired length of the master key. These are then used in a pseudorandom function for a fixed number of iterations. In WinRAR, the salt is stored as an option field in the file header and the key length depends on the file version. WinRAR uses PBKDF2, which implements HMAC with SHA-1 as the pseudorandom function [23].

## 2.3 WinRAR compression

WinRAR uses a proprietary compression implementation developed by Eugene Roshal [17]. This implementation includes several well-known compression algorithms such as: Lempel-Ziv-Storer-Szymanski (LZSS), PPM with Information Inheritance (PPMII), Intel IA-32 and delta encoding. These methods will be discussed in detail below.

### 2.3.1 LZSS

LZSS is the primary compression method for WinRAR. It is a lossless data compression algorithm derived from LZ77 [24]. LZSS is a dictionary coding technique that utilizes previously seen text as a dictionary. A string of symbols,  $S$ , is replaced by pointers to substrings of  $S$  in the dictionary along with the length of the substring. The pointers are *original* if they point to a substring of the original source. Similarly,

a *compressed* pointer references the compressed representation [24]. The references can be either left- or right- pointing and the scheme allows for recursion.

Storer and Szymanski's scheme addresses a flaw in the original LZ77 algorithm. LZ77 would occasionally generate a reference longer than the target string, resulting in poor compression. To correct this, LZSS omits references that are longer than a specific point. This scheme also uses one-bit flags to indicate whether the following string of data is the original source or a reference.

### 2.3.2 PPMII

PPMII was integrated into WinRAR as of version 2.9 to further reduce compression ratios [21]. PPMII was developed by Dmitry Shkarin as an improvement to the Prediction by Partial Matching model [25, 26]. Broadly, the  $n^{\text{th}}$  symbol of a string is predicted based on the previous  $n - 1$  symbols. The compression of a string is defined by code conditional probability distributions and based on the following assumption [25]:

The larger the common (initial) part of contexts  $s$ , the larger (on the average) the *closeness* of their conditional probability distributions.

This notes that the greater number of common characters two strings have, the greater the probability of predicting the  $n^{\text{th}}$  symbol. This is desirable as a higher probability requires fewer bits to encode. To efficiently store the contexts, an  $M - \text{ary}$  tree is utilized. This is particularly efficient if a text consists of large numbers of short strings.

### 2.3.3 Intel IA-32

Intel IA-32 is a compression scheme introduced in response to the observation that database processing correlates with the hardware constraints of storage I/O [27]. It provides lightweight compression and decompression using single instruction, multiple

data (SIMD) commands to optimize database queries. Data is compressed quickly by reducing the the dynamic range of data. This is accomplished by applying a mask, packed shift, and finally stitching the data together.

#### 2.3.4 Delta encoding

This is the second new technique introduced to optimize compression performace in the newest version. Delta encoding encompasses several techniques that stores data as the difference between successive samples [28]. This is an alternative to directly storing the samples themselves. Generally, the first value in the encoded file is equal to the first value in the original data. The subsequent values are equal to the difference between the current and previous value in the input. That is, for an encoded value  $y_n$  with original inputs  $x_n$ :

$$y_n = x_n - x_{n-1} \tag{2.1}$$

This approach is best suited when the values in the original file have only small changes between adjacent symbols. It is therefore ideal for file representation of a signal, but performs poorly with text files and executable code.

### 3. METHODS

This section provides detailed descriptions of the experiments taken to determine information leakage through the compression side-channel. These experiments include an examination of compression ratios for file type and string detection as well as a man-in-the-middle attack exploiting the independence between the compression and encryption algorithms. The experiments were chosen based on their focus on the compression side-channel and the practicality of implementing the attacks with limited knowledge of an archive’s contents. Unless otherwise indicated, experiments are performed using WinRAR v5.10

#### 3.1 Compression ratios

This experiment is based on work by Kelsey and Polmirova-Nickolova [11,12]. It is run to test the hypothesis:

**Hypothesis 1** The compression of different file types under RAR and RAR5 archives will produce distinct compression ratios.

If this hypothesis holds true, an attacker can make an educated guess as to the contents of an encrypted archived file. This knowledge is useful for identifying file type even when a user applies obfuscating measure such as renaming a file. The information needed to calculate the compression ratio can be obtained by inspecting the file header. Once this information is obtained, the compression ratio can be calculated as shown in Equation 3.1.

The files used in this test are retrieved from the Canterbury Corpus and Maximum Compression benchmark [29,30]. The files included in these collections are selected to give compression results typical of commonly used files. In particular, the Canterbury

Corpus is the main benchmark to test compression algorithms. Details describing the contents of the collections can be found in Appendix A.

The files are categorized into four types as outlined in the Maximum Compression collection: text, executable, graphic, and other. Text file formats include plain text files in English. Executables are Windows executable files such as `.exe` extensions. Graphic files are various image file types. Other types include any files not included under the other categories such as Microsoft Office documents, Adobe PDF or help files. The corpa include only two graphic files of `.jpeg` and `.bmp` types. To increase sample size and provide a wider range of file formats under the graphic file type, additional `.png` and `.gif` formats are included.

All files in the collection were compressed with and without encryption using both RAR and RAR5 file types. For testing purposes, the password “P4ssw0rd” was used for all encrypted archives. The compression ratio,  $c$ , for each archive with packed archive size  $x$  and unpacked size  $y$  is calculated as follows:

$$c = \frac{x}{y} \tag{3.1}$$

The experiment is set up as a Block design with the file types as treatments and encryption and archive format as blocks. This allows the compression ratio of file types to be compared while controlling the variation due to different methods. Analysis of Variance (ANOVA) is then employed to test the existence of a statistical difference between treatments. These tests are carried out at the  $\alpha = 5\%$  significance level.

### 3.2 File detection

The file detection experiments are inspired by the String Presence Detection attacks outlined by Kelsey [12]. This experiment will test the following two hypotheses:

**Hypothesis 2a** Given an uncompressed plaintext string  $S$  and a known file from an encrypted archive, an attacker can determine whether  $S$  appears frequently within the archive.

Table 3.1.  
Sample of compression ratio data.

File Type	RAR, no password	RAR5, no password	RAR, password	RAR5, password
Text	.247	.247	.247	.248
Executable	.356	.356	.356	.356
Other	.490	.491	.490	.491

**Hypothesis 2b** Given an encrypted archive, the compression ratio of the archive and the contained files are correlated.

Suppose that an adversary wants to discover whether a particular file is present in an encrypted compressed archive. He chooses a string,  $S$ , that he knows to occur frequently within the file. If **Hypothesis 2a** holds true, frequent appearances of a string  $S$  from a file imply that the file is likely contained within the archive. If **Hypothesis 2b** holds true, the correlation between the desired file and the archive can suggest whether the file is present.

Kelsey presents a partial known input attack as follows [12]:

1. Given a string,  $S$ , and a known part of a set of messages, the attacker looks for appearances of substrings of  $S$  in the known part of the message.
2. The appearance of substrings of  $S$  is correlated with the compressed length of the message.
3. The attacker determines whether  $S$  appears frequently in the message.

The file *FP.log* is the file to be detected for this experiment. It contains many repetitive strings, which makes it ideal to use for detection. From this file, the following string is chosen:

*compatible; MSIE 5.0; Windows 98*

This string appears 9288 times throughout *FP.log*. This is an extremely high rate of occurrence for a string of this length. Examination of other text files in the compression corpa shows that repetitions of strings of length greater than five is rare. Table 3.2 provides the greatest number of repetitions for strings of various lengths for typical text files in the corpa.

Table 3.2.  
Number of repetitions of text strings of indicated length.

<b>File</b>	<b>8-word</b>	<b>7-word</b>	<b>6-word</b>	<b>5-word</b>
alice29	4	4	6	19
asyoulik	6	7	17	22
fields	4	4	4	4
grammar	2	2	2	4
lcet	5	7	10	10
plarbn	2	3	3	4
xargs	0	2	2	4

Nine other text files of varying sizes and contents were selected from the collection. The ten files were then used to construct 120 encrypted archives each containing three files. In each archive, one file is assumed to be known. The appearances of substrings of  $S$  are then counted for each known file. The number of substring appearances is then compared to the compressed archive length using linear regression to determine if a correlation exists.

For the second half of the experiment, the compression ratio for the encrypted archive is compared with the compression ratio of the file in question. The two-tailed  $t$ -test is used to determine whether the archive's compression ratio is equal to the file's compression ratio. The following formula is used to calculate the  $t$ -value:

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} \quad (3.2)$$

Where  $\bar{x}$  is the average archive compression ratio,  $\mu_0$  is the file compression ratio,  $s$  is the sample standard deviation and  $n$  is the sample size. The  $t$ -critical value,  $t_{\alpha,df}$ , can be calculated using statistical software for comparison. If the calculated  $t$ -value is less than the critical value, then the null hypothesis of  $\bar{x} = \mu_0$  can be said to hold true.

### 3.3 Man-in-the-Middle attack

This attack exploits the independence between the encryption and compression algorithms. It was first introduced by Kohno as an attack against WinZip and later verified by Yeo and Phan [10], [9]. Assume that two users, Alice and Bob, wish to send a secret message in an encrypted compressed archive. Eve is a third individual who wants to discover the content of the secret archive. The attack as outline by Yeo and Phan proceeds as follows:

1. Alice compresses and encrypts `Secret.txt` into `Secret.rar` using compression method 1 and shares the archive with Bob.
2. Eve intercepts `Secret.rar` and modifies the indicated compression method in the file header to compression method 2. She sends this modified archive, say `Secret-prime.rar`, on to Bob.
3. Bob, unaware of Eve's actions, attempts to decompress `Secret-prime.rar` with his secret password. This results in an incomprehensible file, `Corrupted-Secret.txt`. He sends `Corrupted-Secret.txt` to Alice in an attempt to understand what is wrong.
4. Eve again intercepts communication to obtain `Corrupted-Secret.txt`. She then re-compresses `Corrupted-Secret.txt` using compression method 2 to obtain `Unencrypted-Secret.rar`.
5. Finally, Eve modifies the compression method in `Unencrypted-Secret.rar` to method 1. She then decompresses the archive to recover the original `Secret.txt`.



This attack requires modification of the archive's file header. This can be accomplished with a hex editor. For the purposes of this paper, the author used HxD HexEditor to test the attack [31]. This is an opensource hex editor that provides several useful features such as built-in calculation of CRC32.

When carrying out this attack, the original authors noted that there may be an issue in Step 3 when Bob attempts to decompress the modified file. WinRAR will return an error that the CRC check failed and the decompressed file will be automatically deleted. The authors suggest using an unerase utility such as Norton's Unerase Utility to recover the lost file. However, this can be prevented using a built-in feature of WinRAR. When indicating the file path to extract the archive into, the user can simply check the "Keep broken files" option under miscellaneous. The user will still receive an error, but the extracted file will be saved where indicated. An internet search shows that CRC checksum errors are common when extracting archives and this is a frequently used method.

There are two important notes involving the modification of the compression method. First, the compression may be changed to any of six possible methods. However, modifying the method without altering the packed and total file sizes will result in some loss of file contents. It is also difficult to accurately predict the correct compression ratio of various methods. To circumvent this issue, the compression method is set to 0x30, which indicates no compression. The total file size is then modified to equal the packed file size which requires no extra calculations on the part of the attacker.

Secondly, the choice of no compression is important to preserving the contents of the original file. In Step 2, Bob must enter his secret password in order for WinRAR to proceed with decryption and decompression of the file. WinRAR first decrypts the contents of the archive before attempting decompression. Recall that Eve has modified the file header within the archive to indicate that there is no compression on the file. This results in WinRAR outputting the exact contents contained in the archive. The file that is obtained from this step is the version of the original

`Secret.txt` compressed using compression method 1. Due to the compression, the file appears incomprehensible to Bob, who was expecting a decompressed file. This `Corrupted-Secret.txt` is all that is needed for Eve to reconstruct `Secret.txt`.

Eve's choice of the compression method in Step 4 is significant to the success of the attack. Adding `Corrupted-Secret.txt` to a WinRAR archive using no compression ensures that the archive contains a copy of the original file under compression method 1. If another method is applied, the archive will contain two layers of compression on the file and subsequent attempts at decompression will result in `Corrupt-Secret.txt` as opposed to the desired `Secret.txt`. Eve can obtain the original text by modifying the compression method field in the header to compression method 1. When the archive is unpacked, WinRAR will then use compression method 1, which matches the compression on `Corrupted-Secret.txt` and the file will successfully be recovered.

This attack is tested using both WinRAR v5.0 and v3.42 for verification. The RAR filetype is tested on both WinRAR v3.42 and v5.10 while the new RAR5 format is tested only on version 5.10. Due to differences in file format, the modification of the file header is slightly difference between versions. Information discussing the identification of header information for RAR file types is outlined in Appendix B. Deeper discussion of the RAR5 format is in the following section.

### 3.3.1 RAR5 file header

New to the RAR5 format is the use of variable integers as data types in the header information. Previous versions of WinRAR use unsigned integer values. Variable length quantities allow for the storage of larger values. It also adds slightly more work for an adversary to modify the file header. However, this should not be relied on to increase the security of the archive.

In a *variable integer*, the lowest 7 bits of each byte contain the integer data while the highest bit is a continuation flag. 1 indicates that further bytes are present in the

sequence while 0 indicates the final byte. RAR5 has a maximum of 10 bytes used to represent an integer [32].

To convert the decimal numbers to a variable length quantity, the following steps must be done:

1. Represent the decimal value in binary notation.
2. Beginning with the least significant bit, divide the binary number into into groups of 7 digits. If a group has fewer than 7 bits available, pad with 0.
3. Append 0 to the beginning of the lowest 7 bits to indicate the end of the integer. Append 1 to the other groups of 7.

```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 52 61 72 21 1A 07 01 00 C9 70 AF E8 0C 01 05 08 Rar!....Ép`è....
00000010 00 07 01 01 86 91 83 80 00 FE 64 75 B4 29 02 03 ....t`f€.pdu')..
00000020 0B C7 90 03 04 99 A4 09 20 BA 7D 00 66 80 0B 00 .Ç...µ*. °}.f€..
00000030 0B 61 6C 69 63 65 32 39 2E 74 78 74 0A 03 02 80 .alice29.txt...€

```

Fig. 3.1. A RAR5 archive with packed size and compression information highlighted

It is most convenient to modify the total file size field first. The total and packed file sizes can be obtained through the "info" option in WinRAR. Convert both sizes to variable length quantities as outlined above and replace the total file size field with the appropriate integer.

Next, the two bytes containing compression information can be located relative to the Host OS field. According to the WinRAR technote [32], the compression method immediately precedes the Host OS field. First, convert the original bytes to binary. Bits 8-10 define the compression method. In decimal form, 0 indicates no compression while 5 is best compression. Modify the bits as necessary. Convert the binary number back to hexadecimal and replace the fields as necessary.

## 4. RESULTS

In this section, the results of the experiments described in the previous section are discussed.

### 4.1 Compression ratios

The first input for the experiment allows each file to be considered a separate treatment. The compression and password options are then considered blocks, of which there are four total. The files are divided into four groups: Text, Executable, Graphics and Other. These are encoded as treatments 1, 2, 3 and 4, respectively. To balance the experiment, four files for each type are randomly selected and each file is tested in every block. An ANOVA test is run to compare the means of the four treatments at a significance level of  $\alpha = .05$ . A box-plot and basic descriptive statistics of the data follow in Figure 4.1 and Table 4.1.

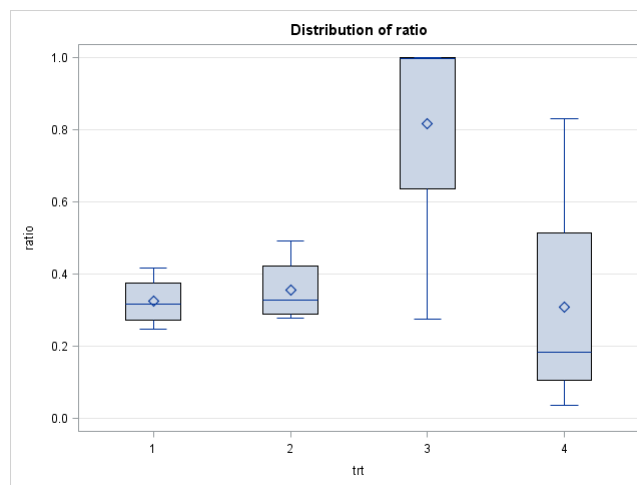


Fig. 4.1. Box Plot of the distributions of different file types.

Table 4.1.  
Descriptive statistics for compression ratio data

Treatment	N Obs	Mean	Std Dev	Minimum	Maximum
1	16	0.3244542	0.0637309	0.2470714	0.4163709
2	16	0.3561775	0.0856055	0.2768610	0.4914785
3	16	0.8181292	0.3235247	0.2754676	1.0009069
4	16	0.3092772	0.3171904	0.0360954	0.8311475

Notice in Figure 4.1, the plots of the treatment means overlap. This suggests that they are not necessarily distinct. To determine whether there exists a significant difference between file types, hypothesis testing on  $H_0$ : *The treatment means are equal* is conducted using Analysis of Variance. SAS provides the ANOVA table in Table 4.2. The P-value of  $< 0.0001$  is less than the stated significance value. Therefore, there is statistical evidence to reject  $H_0$  and the conclusion is that there exists a difference in compression ratios of different file types.

Table 4.2.  
ANOVA table for comparing compression ratios of different file types

Source	DF	Type III SS	Mean Square	F-Value	P-value
trt	3	2.87792404	0.95930801	16.82	$<.0001$
blk	3	0.00000060	0.00000020	0.00	1.0000

To formally test the difference between means, Tukey's comparison for treatment means is implemented. All possible pairs from the data are tested, which make Tukey's comparison most appropriate. Means with the same letter are not considered significantly different. As illustrated in Table 4.3, treatments 2, 4 and 1 are not significantly different. These treatment types correspond to text, executable and

other data files respectively. Graphics are noted to have a mean significantly higher than other file types.

Table 4.3.  
Tukey's comparison of treatment means

<b>Tukey Grouping</b>	<b>Mean</b>	<b>N</b>	<b>trt</b>
A	0.81813	16	3
B	0.35618	16	2
B	0.32445	16	4
B	0.30928	16	1

Finally, Table 4.4 provides 95% confidence intervals for the different file type ratios. These intervals have a 95% chance of containing the true population mean. Investigators with a known compression ratio falling within one of these intervals can assume that the files contained in the archive are of the indicated file type.

Table 4.4.  
95% Confidence Intervals for different file type compression ratios

<b>File Type</b>	<b>Mean</b>	<b>95% Confidence Interval</b>	
Text	0.32445	0.29049	0.35841
Executable	0.35618	0.31056	0.40179
Graphic	0.81813	0.64574	0.99052
Other	0.30928	0.14026	0.47830

## 4.2 File detection

Two experiments are run in this section. The first tests whether the appearance of substrings in the known part of an archive correlates with the compressed length

of the archive. The second experiment tests whether the compression ratio of the archive is correlated with the compression ratio of a file in question.

#### 4.2.1 Appearance of substrings

The archives are constructed as described in Section 3.2. The goal is to identify archives that contain *FP.log* through the appearance of substrings of a string  $S$  in a known file. Archives containing *FP.log* are sorted from the collection. Appearance of substrings are counted for each archive. Linear regression is then applied to determine the correlation between the number of appearances and the compressed size of the archive.

Table 4.5.

SAS output of correlation between size and appearance of substrings where the file is present

<b>Root MSE</b>	495032	<b>R-Square</b>	0.2520
<b>Dependent Mean</b>	1293068	<b>Adj R-Sq</b>	0.1273
<b>Coeff Var</b>	38.28347		

Table 4.6.

SAS output of correlation between size and appearance of substrings where the file is not present

<b>Root MSE</b>	317309	<b>R-Square</b>	0.1396
<b>Dependent Mean</b>	109798	<b>Adj R-Sq</b>	0.0614
<b>Coeff Var</b>	288.99243		

Tables 4.5 and 4.6 show the SAS output for the correlation values. The model uses multiple linear regression, so the **Adj R-sq** is the most appropriate statistic. Notice that  $R_{present}^2 = 0.1273$  and  $R_{notpresent}^2 = 0.0614$ . This implies that the correlation is

stronger for archives that do contain the file in question. This supports the hypothesis that an attacker can determine whether a string  $S$  appears frequently within an archive.

#### 4.2.2 Difference of ratios

The same collection of archives utilized in Section 4.2.1 are examined again. For this experiment, the compression ratios of *FP.log* and the full archive are compared to discover if there is a difference in their average. For this experiment, it is not necessary to have a known file from the archive. The two-tailed  $t$ -test as shown in Equation 3.2 is implemented. All necessary statistical computations were performed using SAS statistical software.

For archives containing *FP.log*, the following values are found:

$$\bar{x} = 0.05629, s = 0.01879, n = 36, |t_{.025,35}| = 2.03011$$

For archives that do not contain this file, the values are as follows:

$$\bar{x} = 0.28075, s = 0.05383, n = 84, |t_{.025,83}| = 1.98896$$

The  $t$ -values for each list can then be computed using  $\mu_0 = 0.04334$ .

$$t_{present} = \frac{0.05629 - 0.04334}{0.01879/\sqrt{36}} = 4.135 \quad (4.1)$$

$$t_{notpresent} = \frac{0.28075 - 0.04334}{0.05383/\sqrt{84}} = 40.422 \quad (4.2)$$

Notice that the null hypothesis of  $\bar{x} = 0.04334$  would be rejected in both cases because the calculated  $t$ -values are both larger than their respective critical values. This may be due to a poor choice in significance level. Other levels of  $\alpha$  are shown in Table 4.7. By increasing the confidence of the test, it is possible to differentiate between archives that contain the file under investigation.



Table 4.7.  
Hypothesis testing results for different levels of

	$t_{35}$	$t_{83}$	File present conclusion	File not present conclusion
.01	2.7238	2.6364	reject $H_0$	reject $H_0$
.001	3.5912	3.4116	reject $H_0$	reject $H_0$
.0001	4.3888	4.08569	fail to reject $H_0$	reject $H_0$

### 4.2.3 Man-in-the-Middle

The attack described in Section 3.3 is tested using WinRAR v3.42 and v5.10. The file *alice29.txt* is used for testing in all cases. The attacks on RAR and RAR5 formats are discussed separately below.

#### RAR file format

The first step of the attack requires changing the compression method and total file size in the file header. As discussed in Section 3.3, setting the compression method to no compression, denoted by `0x30`, provides the best results. Additionally, the total file size is altered to equal the packed file size in the header. This step is illustrated in Figures 4.2 and 4.3 below.

```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
00000000 52 61 72 21 1A 07 00 CF 90 73 00 00 0D 00 00 00 00 00 00 00 00 B8 9E 74 44 Rar!...İ.s.....,žtD
00000018 94 35 00 20 C8 00 00 19 52 02 00 02 BA 7D 00 66 6F 06 3A 21 1D 33 0B 00 "5. È...R...°}.fo.:!3..
00000030 20 00 00 00 61 6C 69 63 65 32 39 2E 74 78 74 86 D1 5B 0F 77 F6 0A BC 00 ...alice29.txt+N[.wö.4.
00000048 C0 51 EC 7C F9 C5 FC E1 BF D7 35 EA 3D 6F DF 3A 37 92 4B F9 D5 C1 74 58 ÀQi|úÁúá¿*5è=oS:7'KúÓAtX
00000060 B7 D9 0C 9E E3 C0 B4 96 EB 06 58 4B 37 44 B9 AF 62 65 5A 68 D5 6C 79 92 ·Û.žžÄ'~è.XK7D^`beZhÓly'
00000078 7F 22 3E E5 98 FB 94 0C 3D 46 88 A0 98 09 6A 17 9A 5E AB 8B 56 67 8C E5 .">â~û".=F^ ".j.š^«VgĚâ

```

Fig. 4.2. The original *alice29.rar* archive with the compression method circled and the total file size inside the rectangle

```

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
00000000 52 61 72 21 1A 07 00 CF 90 73 00 00 0D 00 00 00 00 00 00 00 B8 9E 74 44 Rar!...İ.s.....,žtD
00000018 94 35 00 20 C8 00 00 20 C8 00 00 02 BA 7D 00 66 6F 06 3A 21 1D 30 0B 00 "5. È.. È...°).fo.:!0..
00000030 20 00 00 00 61 6C 69 63 65 32 39 2E 74 78 74 86 D1 5B 0F 77 F6 0A BC 00 ...alice29.txt†Ñ[.wö.4.
00000048 C0 51 EC 7C F9 C5 FC E1 BF D7 35 EA 3D 6F DF 3A 37 92 4B F9 D5 C1 74 58 ÀQi|ùÄú¿×5ê=oS:7'KüÖÄtX
00000060 B7 D9 0C 9E E3 C0 B4 96 EB 06 58 4B 37 44 B9 AF 62 65 5A 68 D5 6C 79 92 ·Ü.žšÄ'-ë.XK7D²`beZh01y'
00000078 7F 22 3E E5 98 FB 94 0C 3D 46 88 A0 98 09 6A 17 9A 5E AB 8B 56 67 8C E5 .">â"û".=F" ".j.š"«<VgQâ

```

Fig. 4.3. The modified *alice29-prime.rar* with the compression method circled and the total file size inside the rectangle

The decompression of *alice29-prime.rar* results in what looks like garbage text. In reality, it is the unencrypted compressed version of the original file. The remaining challenge is to reconstruct the original file given the corrupted text. The final step of the attack outlined in [9] and [10] is to re-compress *alice29-corrupted.txt* using compression method 0x30, restore the compression method and total file size to their original values and decompress the archive.

The attack fails as outlined during the final step. Comparison of the file contents with an unencrypted compressed version of the original file verify that the encryption has been removed. However, neither WinRAR v3.42 nor v5.10 is capable of decompressing the resulting archive correctly.

In addition to the steps outlined in the original attack, the author made several modifications to the final archive. The modifications were based on the following observations.

1. There is now padding at the end of the file.
2. The encrypted packed archive size is larger than the unencrypted packed archive size.
3. The RAR version needed to extract the file has changed.

To address these issues, supplementary modifications can be done. None of the new changes require any additional knowledge on the part of the attacker. First, the zero padding at the end of the file is deleted. The number of bytes in the padding

is then subtracted from the packed archive size in the file header. Next, the `UNP_VER` field is adjusted to reflect the same version as indicated in the original *alice29.rar* archive. To reduce errors, the file and header CRC32 is recalculated using the built in HxD function. The corresponding fields are also updated. With these changes, it is possible to fully recover the original file.

### **RAR5 file format**

The RAR5 format is tested in WinRAR v5.10 using the same test file and attack outline. When implementing the attack, the new format requires further calculations to modify the required fields. The variable length quantities for the packed archive size and the compression method are calculated as described in Section 3.3.1. This section illustrates the steps necessary to modify the compression method as an example to readers.

In the encrypted compressed archive *alice29.rar*, the compression method is represented by the hexadecimal numerals `0x800B`. This is represented in binary notation as `10000000 0001011`. The final three digits in the binary string represent the compression method used in the archive. Currently compression method 3, normal compression, is selected. To move forward with the attack, compression method 0 will be applied by changing the digits to obtain the string `10000000 0000000`. The fourth digit indicates the dictionary size required to extract data. Since there is no compression in the archive, this bit is not necessary. Finally, converting back to hexadecimal produces a final value of `0x8000`. This is used to replace the initial compression method.

In contrast to the earlier file formats, the attack fails at this point. Despite changes to the file header, the RAR5 format is capable of extracting the original contents with no issue. The extraction does not result in CRC checksum errors as the previous versions do. Without the extraction of corrupted contents, the attack is unable to

proceed. Subsequent attempts to sabotage the file header to force an error failed. The fields altered included CRC32 fields, file flags, and attributes.

## 5. SUMMARY

The findings presented in Section 4 include several novel results. These will be discussed in detail followed by a brief suggestion of countermeasures to prevent information leakage.

### 5.0.4 Discussion

In Section 4.1, statistical methods show that it is possible to distinguish different file types based on an archive's compression ratio. Therefore, the proposed **Hypothesis 1** holds true. It is important to notice that, as illustrated in Table 4.3, Text, Executable and Other compression ratios are not distinct. However, graphic files consistently compress at a ratio considerably higher than other file types. This is likely due to the fact that many image formats implement some form of compression [33]. If the data in an archive has already been compressed, WinRAR's algorithms can do little to further reduce an archive's size. This results in a packed file size very close to the total file size.

This attack is most effective if an investigator is considering compression ratios to assist in identifying whether an archive contains images. For example, in child pornography cases a forensic investigator may need to identify archives with large amounts of images. Compression ratio inspection provides a simple method of identification for archives with these types of contents. The information necessary is very minimal and can be found from any archive which makes this attack easy to implement in a variety of situations. Table 4.4 provides some intervals to be used for identifying file types. Generally, an archive with a compression ratio greater than .64 can reasonably be assumed to contain images. The ability to identify file types within an archive

helps save valuable time and effort that could potentially be lost in attempting to crack archives with irrelevant contents.

The appearance of substring experiment in Section 4.2.1 supported the hypothesis that substrings in the known part of an archive correlate with the compressed size of the archive. This correlation is likely due to the general compression scheme utilized by WinRAR. If a file is present in an archive, the appearance of substrings will allow both LZSS and PPMII compression schemes to work more efficiently. In turn, this results in a lower packed size for the archive. This provided the most surprising results of the experiments as the conclusions were not immediately obvious from the raw data.

This attack does have several drawbacks. First, the file selected for examination has an extremely high number of repeated strings as stated in Section 3.2. Unless an investigator is looking for similarly structured files, such as log files, it is unlikely that typical text will include similar levels of repetition. This would result in a weaker effect on the overall compression size which may cause the correlation to become too weak for detection. However, the appearance of substring attack is ideal to identify files containing profile or bank account information that include many repeated fields.

Secondly, a relatively large collection of archives was examined, which strengthened the power of the statistical process. A collection of this size may not be available for study. Finally, the archives containing the file were known ahead of time. While this experimental design is sufficient to show correlation, it is not practical to execute on completely unknown data. For future testing, a Monte Carlo experiment may provide more accurate results for modeling the relationship between substrings and archive size.

Section 4.2.2 showed that, with sufficiently significant levels of  $\alpha$  it is possible to distinguish archives that contain a file from those that don't. Adequate evidence is given to show that **Hypothesis 2b** is valid. It should be noted that in this experiment the ratios of archives containing the file have a significantly different average than those that don't. In the event that the averages are closer in value, the

author suggests that lower values of  $\alpha$  will be capable of distinguishing between them. This attack is ideal to use on files that are highly compressible as its compression ratio will have a significant effect on that of the archive. The selection of files most suited to this attack suffers from the same issues outlined for the appearance of substrings attack.

Both the appearance of substrings and difference of ratios attacks can extend their usefulness in exfiltration detection measures. For example, if numerous archives are detected leaving an organization's system, sensitive information such as client data can be checked against the archives as outlined. This can provide a reasonable perception of what information has been compromised.

Finally, experiments with the Man-in-the-Middle attack in Section 4.2.3 provided suggestions for improvement. Despite claims that the original attack is capable of obtaining the plain text of a file in an archive, it does not perform as suggested. Following the attack as outlined in the literature will result in the removal of encryption from an archive. However, the compressed file is still unintelligible.

To remedy this, the author suggests some variation in the final step of the attack. First, the files tend to accumulate extra padding at the end. This is simple to identify as it consists of a string of hexadecimal values `0x00`. The padding may be generated from the loss of the password and salt after the encryption is removed. To avoid conflict with the file size, the packed file size needs to be adjusted according to the amount of padding removed. Secondly, WinRAR uses standard CRC32 checksums, which can be computed with off the shelf software and applied in the relevant fields. Finally, the unpacking version field should be updated to the value in the original archive's field to avoid compatibility issues. All of the extra information needed can be discovered using the archives that an adversary has access to. These steps will insure that the contents of an encrypted compressed archive can be revealed. The attack has been verified on **RAR** archives using both WinRAR v3.42 and v5.10.

Despite the success of the revised implementation, **RAR5** formatted archives remain robust against the attack. This is possibly due to the enhanced archive recovery

capabilities in v5.x. The software is more capable of detecting and mitigating changes to file information. Another potential pitfall in attacking the newest file format is the new checksum algorithms. The CRC32 and BLAKE2 checksums are now password dependent. Without knowing the password, it is not feasible to calculate the values necessary in the final step of the attack. However, the older file format is the default method for the newest version and remains very widely used. The attack introduced in this paper is relevant to current information security needs.

### 5.0.5 Countermeasures

In response to the information discovered through the experiments, there are suggestions to circumvent some of the attacks. Aside from the appearance of substrings attack, all of the attacks rely on the assumption that the adversary is able to at least view file header information. The default setting in WinRAR only encrypts a file's contents and the header information remains in plaintext. For this situation, the assumption holds. However, users are able to select an option to encrypt file header information along with the file contents. This would mask information such as total and packed file size, compression method and any additional file attributes.

For further security of files, the author suggests using the RAR5 file format when possible. It has the same weakness against the first three attacks as the older file versions. However, it is resilient against the Man-in-the-middle attack. Thus, it provides slightly improved security over previous versions.

### 5.0.6 Conclusion and open questions

This paper shows that knowledge of information in an encrypted archive can be leaked via the study of compression properties. These attacks require less time and computing resources than traditional attacks against the encryption of an archive. Issues in an attack are addressed to create a successful method for recovering archived files. This has been verified with two different versions of WinRAR but the effec-



tiveness with other compression software remains to be evaluated. There is also a possibility of using this attack against an archive containing multiple files. All of the presented methods are efficient for investigators to implement as a first line of query to discover information about an unknown archive. These methods also highlight an area that is lacking in security for the WinRAR software. It is a future challenge to provide a good compression scheme with effectively implemented encryption.

Some open questions remain in relation to the string detection attacks. The effect of string frequency in the appearance of substrings remains open for further investigation. As discussed in Section 5.0.4, the attack is effective on highly compressible files such as logs or databases. However, many text files do not have a high number of repetitive strings. The length of the string may also influence the correlation with an archive's size. Further investigation into the effects of repetition and length remain open.

The experimental design used emphasizes the use of statistics to conclude the validity of a hypothesis. When conducting the literature review, very few papers implemented rigorous statistical methods to reach conclusions. The meaning of data can be counter-intuitive and it is possible to reach incorrect conclusions without proper analysis. The author encourages future researchers to use experimental methods to provide strong validity for information security research.

## REFERENCES

## REFERENCES

- [1] Symantec. Trojan.Dropper Technical Details. [Online]. Available: [http://www.symantec.com/security\\_response/writeup.jsp?docid=2002-082718-3007-99tabid=2](http://www.symantec.com/security_response/writeup.jsp?docid=2002-082718-3007-99tabid=2)
- [2] WinZip. What can I do if I forget the encryption password for my zip file? [Online]. Available: <http://kb.winzip.com/kb/entry/79/>
- [3] J. Chen, J. Zhou, K. Pan, S. Lin, C. Zhao, and X. Li, “The security of key derivation functions in WINRAR,” *Journal of Computers*, vol. 8, no. 9, pp. 2262–2268, 2013.
- [4] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, and A. Shamir, “Key recovery attacks of practical complexity on AES variants with up to 10 rounds,” *IACR eprint server*, vol. 374, 2009.
- [5] A. Biryukov and D. Khovratovich, “Related-key cryptanalysis of the full AES-192 and AES-256,” in *Advances in Cryptology-ASIACRYPT 2009*. Springer, 2009, pp. 1–18.
- [6] H. Demirci and A. A. Selçuk, “A meet-in-the-middle attack on 8-round AES,” in *Fast Software Encryption*. Springer, 2008, pp. 116–126.
- [7] J. Lu, O. Dunkelman, N. Keller, and J. Kim, “New impossible differential attacks on AES,” in *Progress in Cryptology-INDOCRYPT 2008*. Springer, 2008, pp. 279–293.
- [8] A. Biryukov, D. Khovratovich, and I. Nikolić, “Distinguisher and related-key attack on the full AES-256,” in *Advances in Cryptology-CRYPTO 2009*. Springer, 2009, pp. 231–249.
- [9] G. S.-W. Yeo and R. C.-W. Phan, “On the security of the WinRAR encryption feature,” *International Journal of Information Security*, vol. 5, no. 2, pp. 115–123, 2006.
- [10] T. Kohno, “Attacking and repairing the WinZip encryption scheme,” in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 72–81.
- [11] D. Polimirova-Nickolova and E. Nickolov, “Examination of archived objects’ size influence on the information security when compression methods are applied,” in *Third International Conference Information Research, Applications and Education*, 2005, p. 130.
- [12] J. Kelsey, “Compression and information leakage of plaintext,” in *Fast Software Encryption*. Springer, 2002, pp. 263–276.

- [13] L. Ji-Zhong, J. Lie-Hui, Y. Qing, and X. Yao-Bin, "Hybrid method to analyze cryptography in software," in *Multimedia Information Networking and Security (MINES), 2012 Fourth International Conference on*. IEEE, 2012, pp. 930–933.
- [14] C. Maartmann-Moe, S. E. Thorkildsen, and A. Årnes, "The persistence of memory: Forensic identification and extraction of cryptographic keys," *digital investigation*, vol. 6, pp. S132–S140, 2009.
- [15] G. Fellows, "WinRAR temporary folder artefacts," *Digital Investigation*, vol. 7, no. 1, pp. 9–13, 2010.
- [16] D. Gupta and B. M. Mehtre, "Recent trends in collection of software forensics artifacts: Issues and challenges," in *Security in Computing and Communications*. Springer, 2013, pp. 303–312.
- [17] RarLab. WinRAR at a glance. [Online]. Available: <http://www.winrar.com/website/index.php?id=features>
- [18] ——. WinRAR - what's new in the latest version. [Online]. Available: <http://www.rarlab.com/rarnew.htm>
- [19] WinRAR, "User's manual: Rar 5.10 console version," 2014.
- [20] J. S. Plank, K. M. Greenan, and E. L. Miller, "Screaming fast galois field arithmetic using intel simd instructions." in *FAST*, 2013, pp. 299–306.
- [21] WinRAR, "What's new in the latest version - version 3.00," 2002.
- [22] N. Standard, "Announcing the advanced encryption standard (AES)," *Federal Information Processing Standards Publication*, vol. 197, 2001.
- [23] M. S. Turan, E. B. Barker, W. E. Burr, and L. Chen, "SP 800-132. recommendation for password-based key derivation: Part 1: Storage applications," National Institute of Standards & Technology, Gaithersburg, MD, United States, Tech. Rep., 2010.
- [24] J. A. Storer and T. G. Szymanski, "Data compression via textual substitution," *Journal of the ACM (JACM)*, vol. 29, no. 4, pp. 928–951, 1982.
- [25] D. Shkarin, "Improving the efficiency of the ppm algorithm," *Problems of information transmission*, vol. 37, no. 3, pp. 226–235, 2001.
- [26] ——. "PPM: One step to practicality," in *Data Compression Conference, 2002. Proceedings. DCC 2002*, 2002, pp. 202–211.
- [27] R. Intel, "Intel 64 and IA-32 architectures optimization reference manual," *Intel Corporation*, May, 2012.
- [28] S. W. Smith *et al.*, "The scientist and engineer's guide to digital signal processing," 1997.
- [29] M. Powell. The Canterbury corpus. [Online]. Available: <http://corpus.canterbury.ac.nz/>

- [30] MaximumCompression. Lossless data compression software benchmarks/comparisons. [Online]. Available: <http://www.maximumcompression.com/>
- [31] M. Hörz, “HxD–HexEditor,” <http://mh-nexus.de/en/hxd/>, 2002–2009.
- [32] RarLab. Rar 5.0 archive format. [Online]. Available: <http://www.rarlab.com/technote.htm>
- [33] M. Prantl, “Image compression overview,” *arPX*.
- [34] WinRAR, “RAR version 3.42 – technical information,” 2004).

## APPENDICES

## A. COMPRESSION CORPA

The following is a description of the files included in the Canterbury Corpus [29] and Maximum Compression [30] compression testing benchmarks.

Table A.1.: Details of compression testing files

File Name	Description
alice29.txt	English text of "Alice in Wonderland"
asyoulik.txt	English text of Shakespeare's "As You Like"
cp.html	HTML source code
fields.c	C source code
grammar.lsp	LISP source code
kennedy.xls	Microsoft Excel spreadsheet
lcet10.txt	English text of Workshop on Electronic Texts proceedings
plrabn12.txt	English text of "Paradise Lost"
ptt5	CCITT test set
sum	SPARC executable
xargs.1	GNU manual page
world95.txt	English text of 1995 CIA World Fact Book
FP.txt	Website traffic log file
english.txt	Alphabetically sorted English word list
AcroRd32.exe	Acrobat Reader 5.0 executable
MSO97.dll	Microsoft Office 97 Dynamic Link Library
rafale.bmp	Bitmap image
A10.jpg	JPEG image
vcfiu.hlp	Delphi First Impression OCX Help file

*continued on next page*

Table A.1.: *continued*

File Name	Description
ohs.doc	Occupational Health and Safety Microsoft Word file
FlashMX.pdf	Macromedia Flash MX manual Adobe Acrobat file
tux.png	PNG image
Nature.gif	GIF image



## B. RAR FILE HEADER

The information presented in this table is based on the WinRAR 3.42 technical note [34].

Table B.1.: RAR file header fields

Field	Length
HEAD_CRC	2 bytes
HEAD_TYPE	1 byte
HEAD_FLAGS	2 bytes
HEAD_SIZE	2 bytes
HEAD_CRC	2 bytes
HEAD_TYPE	1 byte
HEAD_FLAGS	2 bytes
HEAD_SIZE	2 bytes
RESERVED1	2 bytes
RESERVED2	4 bytes
HEAD_CRC	2 bytes
HEAD_TYPE	1 byte
HEAD_FLAGS	2 bytes
HEAD_SIZE	2 bytes
PACK_SIZE	4 bytes
UNP_SIZE	4 bytes
HOST_OS	1 byte
FILE_CRC	4 bytes

*continued on next page*

Table B.1.: *continued*

Field	Length
FTIME	4 bytes
UNP_VER	1 byte
METHOD	1 byte
NAME_SIZE	2 bytes
ATTR	4 bytes
HIGH_PACK_SIZE	4 bytes
HIGH_UNP_SIZE	4 bytes
FILE_NAME	variable size
SALT	8 bytes
EXT_TIME	variable size