On Achieving Encrypted File Recovery

Xiaodong Lin[1], Chenxi Zhang[2], and Theodora Dule[1]

[1] University of Ontario Institute of Technology, Oshawa, Ontario, Canada
{Xiaodong.Lin,Theodora.Dule}@uoit.ca
[2] University of Waterloo, Waterloo, Ontario, Canada
c14zhang@engmail.uwaterloo.ca

Abstract. As digital devices become more prevalent in our society, evidence relating to crimes will be more frequently found on digital devices. Computer forensics is becoming a vital tool required by law enforcement for providing data recovery of key evidence. File carving is a powerful approach for recovering data especially when file system metadata information is unavailable. Many file carving approaches have been proposed, but cannot directly apply to encrypted file recovery. In this paper, we first identify the problem of encrypted file recovery, and then propose an effective method for encrypted file recovery through recognizing the encryption algorithm and mode in use. We classify encryption modes into two categories. For each category, we introduce a corresponding mechanism for file recovery, and also propose an algorithm to recognize the encryption algorithm and mode. Finally, we theoretically analyze the accuracy rate of recognizing an entire encrypted file in terms of file types.

Keywords: Data Recovery, File Carving, Computer Forensics, Security, Block Cipher Encryption/Decryption.

1 Introduction

Digital devices such as cellular phones, PDAs, laptops, desktops and a myriad of data storage devices pervade many aspects of life in today's society. The digitization of data and its resultant ease of storage, retrieval and distribution have revolutionized our lives in many ways and led to a steady decline in the use of traditional print mediums. The publishing industry, for example, has struggled to reinvent itself by moving to online publishing in the face of shrinking demand for print media. Today, financial institutions, hospitals, government agencies, businesses, the news media and even criminal organizations could not function without access to the huge volumes of digital information stored on digital devices.

Unfortunately, the digital age has also given rise to digital crime where criminals use digital devices in the commission of unlawful activities like hacking, identity theft, embezzlement, child pornography, theft of trade secrets, etc. Increasingly, digital devices like computers, cell phones, cameras, etc. are found at crime scenes during a criminal investigation. Consequently, there is a growing need for investigators to search digital devices for data evidence including

X. Lai et al. (Eds.): E-Forensics 2010, LNICST 56, pp. 1–13, 2011.

emails, photos, video, text messages, transaction log files, etc. that can assist in the reconstruction of a crime and identification of the perpetrator. One of the decade's most fascinating criminal trials against corporate giant Enron was successful largely due to the digital evidence in the form of over 200,000 emails and office documents recovered from computers at their offices. Digital forensics or computer forensics is an increasingly vital part of law enforcement investigations and is also useful in the private sector for disaster recovery plans for commercial entities that rely heavily on digital data, where data recovery plays an important role in the computer forensics field.

Traditional data recovery methods make use of file system structure on storage devices to rebuild the device's contents and regain access to the data. These traditional recovery methods become ineffective when the file system structure is corrupted or damaged, a task easily accomplished by a savvy criminal or disgruntled employee. A more sophisticated data recovery solution which does not rely on the file system structure is therefore necessary. These new and sophisticated solutions are collectively known as file carving. File carving is a branch of digital forensics that reconstructs data from a digital device without any prior knowledge of the data structures, sizes, content or type located on the storage medium. In other words, the technique of recovering files from a block of binary data without using information from the file system structure or other file metadata on the storage device.

Carving out deleted files using only the file structure and content could be very promising [3] due to the fact that some files have very unique structures which can help to determine a file's footer as well as help to correct and verify a recovered file, e.g., using a cyclic redundancy check (CRC) or polynomial code checksum. Recovering contiguous files is a trivial task. However, when a file is fragmented, data about the file structure is not as reliable. In these cases, the file content becomes a much more important factor than the file structure for file carving. The file contents can help us to collect the features of a file type, which is useful for file fragment classification. Many approaches [4,5,6,7,8] of classification for file recovery have been reported and are efficient and effective. McDaniel et al. [4] proposed algorithms to produce file fingerprints of file types. The file fingerprints are created based on byte frequency distribution (BFD) and byte frequency cross-correlation (BFC). Subsequently, Wang et al. [5] created a set of modes for each file type in order to improve the technique of creating file fingerprint and thus to enhance the recognition accuracy rate: 100% accuracy for some file types and 77% accuracy for JPEG file. Karresand et al. [7,8] introduced a classification approach based on individual clusters instead of entire files. They used the rate of change (RoC) as a feature, which can recognize JPEG file with the accuracy up to 99%.

Although these classification approaches are efficient, they have no effect on encrypted files. For reasons of confidentiality, in some situations, people encrypt their private files and then store them on the hard disk. The content of encrypted files is a random bit stream, which provides no clue about original file features or useful information for creating file fingerprints. Thus, traditional classification

approaches cannot be directly applied to encrypted file recovery. In this paper, we introduce a recovering mechanism for encrypted files. To the best of our knowledge, this is the first study of encrypted file recovery. Firstly, we categorize block cipher encryption mode into two groups: block-decryption-dependant, and block-decryption-independent. For each group, we present an approach for file recovery. Secondly, we present an approach for recognizing block cipher mode and encryption algorithm. Based on the introduced approach, encrypted files can be recovered. Lastly, we analyze our proposed scheme theoretically.

The rest of the paper is organized as follows. Section 2 briefly introduces problem statement, objective and preliminaries that include file system, file fragmentation, and file encryption/decryption. According to different block cipher encryption modes, Section 3 presents a corresponding mechanism for file recovering. Section 4 introduces an approach of recognizing a block cipher mode and an encryption algorithm. Section 5 theoretically analyzes our proposed approach. Finally, we draw the conclusions of this study and give the future work in Section 6.

2 Preliminaries and Objective

2.1 File System and File Fragmentation

We use the FAT file system as an example to introduce general concepts about file systems. In a file system, a file is organized into two main parts: (1) The first part is the file identification and metadata information, which tell an operating system (OS) where a file is physically stored; (2) The second part of a file is its physical contents that are stored in a disk data area. In a file system, a cluster (or block) is the smallest data unit of transfer between the OS and disk. The name and starting cluster of a file is stored in a directory entry, which presents the first cluster of the file. Each entry of a file allocation table (FAT) records its next cluster number where a file is stored and a special value is used to indicate the end of file (EOF), for example, 0xffffffff as end of cluster chain markers for one of three versions of FAT, i.e., FAT32. As shown in Fig. 1, the first cluster number of file a.txt is 32, and the following cluster number is 33, 39, 40. When a file is deleted, its corresponding entries at the file allocation table are wiped out to zero. As shown in Fig. 1, if a.txt is deleted, the entries, 32, 33, 39, and 40, are set to "0". However, the contents of a.txt in the disk data area remain. The objective of a file carver is to recover a file without the file allocation table.

When files are first created, they may be allocated in disk entirely and without fragmentation. As files are modified, deleted, and created over time, it is highly possible that some files become fragmented. As shown in Fig. 1, a.txt and b.txt are fragmented, and each of them are fragmented into two fragments.

2.2 Problem Statement and Objective

We will now give an example to properly demonstrate the issue we will address in this paper. Suppose that there are several files in a folder. Some files are

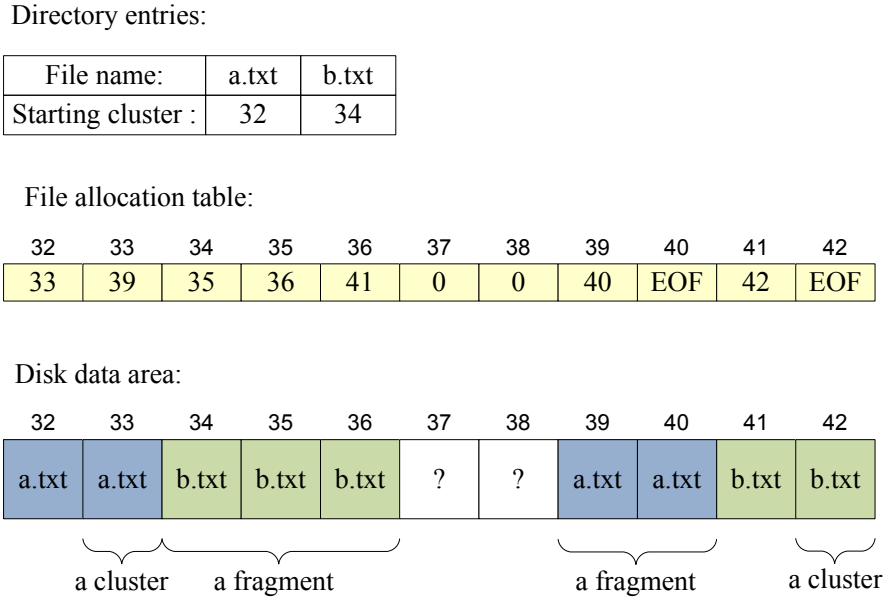

Fig. 1. The illustration of a file system and file fragmentation

unencrypted while some files are encrypted due to some security and privacy reasons. It is worth noting that the encrypted files are encrypted by a user not an operating system. Now assume that all of these files are deleted inadvertently. Our objective is to recover these files, given that the user still remembers the encryption key for each encrypted file.

First of all, let us consider the situation where the files are unencrypted. As shown in Fig. 2(a), file F_1 and F_2, which are two different file types, are fragmented and stored in the disk. In this case, a file classification approach can be used to classify the file F_1 and F_2, and then the two files can be reassembled. The reason why F_1 and F_2 can be classified is that the content features of F_1 and F_2 are different. Based on the features, such as keyword, rate of change (RoC), byte frequency distribution (BFD), and byte frequency cross-correlation (BFC), file fingerprints can be created easily and used for file classification.

However, when we consider the situation where the files are encrypted, the solution of using file classification does not work any more. As illustrated in Fig. 2(b), the encrypted content of files is a random bit stream, and it is difficult to find file features from the random bit stream in order to classify the files accurately. The only information we have is the encryption/decryption keys. Even given these keys, we still cannot simply decrypt the file contents like from Fig. 2(b) to Fig. 2(a). It is not only because the cipher content of a file is fragmented, but also because we cannot know which key corresponds to which random bit stream.

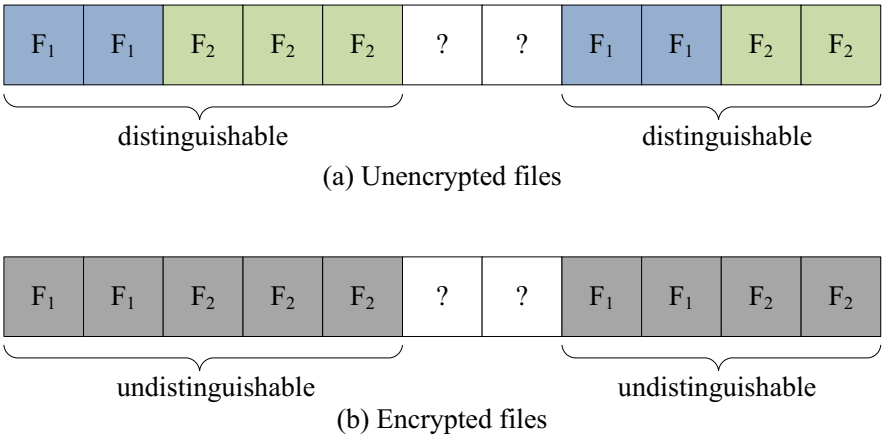

Fig. 2. File F_1 and F_2 have been divided into several fragments. (a) shows the case that F_1 and F_2 are unencrypted, and (b) shows the case that F_1 and F_2 are encrypted.

The objective of this paper is to find an efficient approach to recover encrypted files. Recovering unencrypted files is beyond the scope of this paper because it can be solved with existing approaches.

2.3 File Encryption/Decryption

There is no difference between file encryption/decryption and data stream encryption/decryption. In a cryptosystem, there are two kinds of encryption: symmetric encryption and asymmetric encryption. Symmetric encryption is more suitable for data streams. In symmetric cryptograph, there are two categories of encryption/decryption algorithms: stream cipher and block cipher. Throughout this paper, we focus on investigating the block cipher to address the issue of file carving. There are many block cipher modes of operation in existence. Cipher-block chaining (CBC) is one of the representative cipher modes. To properly present block cipher, we take CBC an example in this subsection.

Fig. 3 illustrates the encryption and decryption processes of CBC mode. To be encrypted, a file is divided into blocks. The size of a block could be 64, 128, or 256 bits, depending on which encryption algorithm is being used. For example, in DES, the block size is 64 bits. If 128- bit AES encryption is used, then the block size is 128 bits. Each block can be encrypted with its previous block cipher and the key. Also, each block can be decrypted with its previous block cipher and the key. The symbol "$\oplus$" in Fig. 3 stands for Exclusive OR (XOR).

3 Encrypted-File Carving Mechanism

For encrypted-file caving, the most important part is to know what block cipher operation mode is used when a file is encrypted. A user intending to recover

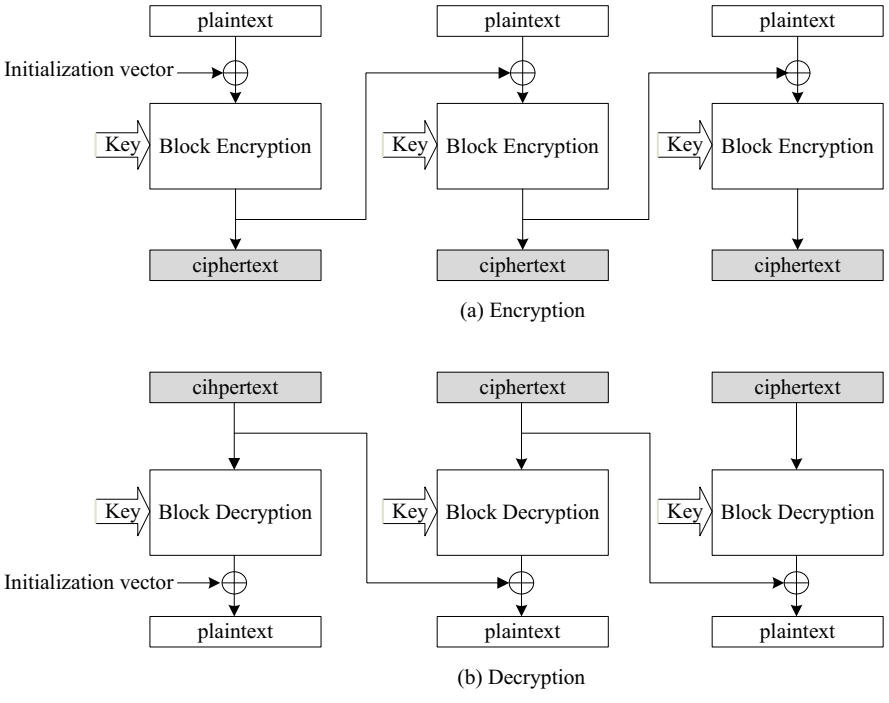

Fig. 3. The encryption and decryption processes of CBC mode

the deleted files may still remember the encryption key, but is unlikely to have any knowledge about the details of the encryption algorithm. In this section, we present a mechanism to recover encrypted files under different block cipher operation modes.

3.1 Recovering Files Encrypted with CBC Mode

In this section, we suppose the file to be recovered is encrypted using CBC mode. From the encryption process of CBC, as shown in Fig. 3(a), we can see that encrypting each block depends on its previous cipher block. As such, the encryption process is like a chain, in which adjacent blocks are connected closely. For example, if we want to get the cipher block i (e.g., $i = 100$), we have to encrypt the plaintext block 1 and get the cipher block 1. Then, we can get the cipher block 2, the cipher block 3, until get the cipher block $i = 100$.

However, the decryption process is different from the encryption process. As shown in Fig. 3(b), to decrypt a cipher block, we only need to know its previous cipher block in addition to the key. For example, if we intent to decrypt the cipher block i (e.g., $i = 100$), we do not have to obtain the cipher block 1 while we only need the cipher block $i - 1 = 99$. We call this feature *block-decryption-independent*.

Based on the block-decryption-independent feature of CBC, we recover an encrypted file according to the following steps.

1. Estimate the physical disk data area where an encrypted file to be recovered could be allocated.
2. Perform brute-force decryption: decrypt each block in the estimated disk data area using the remembered encryption key.
3. Recognize the decrypted fragments, collect the recognized fragments, and reassemble the fragments.

In file systems, the size of a cluster depends on the operating system, e.g., 4KB. However, the size is always larger than and multiple of the size of an encryption block, e.g., 64 or 128 bits. Thus, we can always decrypt a cluster from the beginning of a cluster.

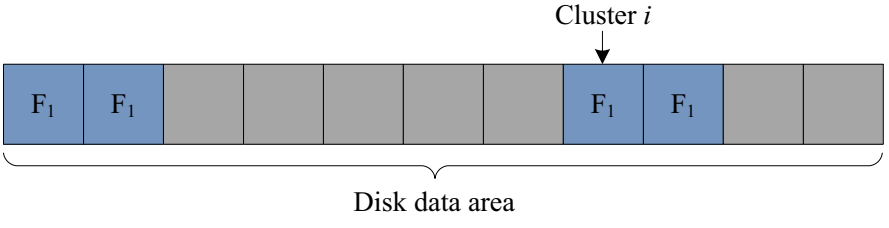

Fig. 4. Decrypted clusters in disk data area

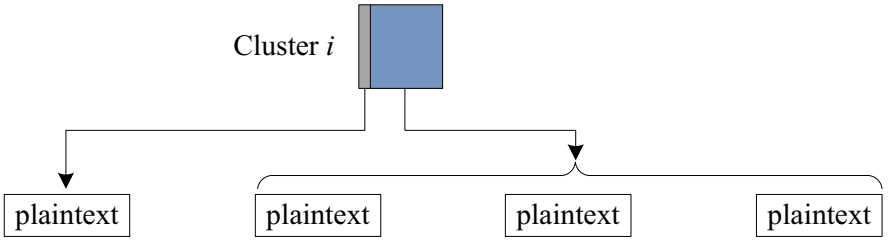

Fig. 5. The first block of Cluster i in Fig. 4 is not decrypted correctly

The encrypted file is a double-edged sword. On the one hand, ciphertext makes us unable to create file fingerprint for file classification. On the other hand, decrypted content makes it easier to classify decrypted file in the disk data area. For example, suppose we intent to recover the file F_1 in Fig. 2(b), and we know the encryption key, K. Using key K, we perform decryption on all clusters. The decrypted clusters of F_1 are shown in Fig. 4. For the clusters that are not part of F_1, the decryption can be treated as encryption using key K. Hence, the clusters that are not parts of F_1 become random bit streams,

which are presented using gray squares in Fig. 4. The random bit streams have no feature of a file type and thus decryption is helpful for us to classify the fragments of F_1 from the disk data area.

Since F_1 is fragmented, cluster i in Fig. 4 cannot be decrypted completely. However, only the first CBC block in cluster i is not decrypted correctly, and the blocks following cluster i can be decrypted correctly according to the block-decryption-independent feature of CBC mode, shown in Fig. 5. This fact does not affect file classification because a block size is far smaller than a cluster size. It is worth noticing that we adopt the existing classification approaches [4,5,6,7,8] for file carving in the file classification process (Step 3). Designing a file classification algorithm is beyond the scope of this paper.

3.2 Recovering Files Encrypted with PCBC Mode

For block cipher, in addition to CBC mode, there are many other modes. Propagating cipher block chaining (PCBC) is another representative mode. The encryption and decryption processes of PCBC mode are shown in Fig. 6. Let C denote a block of cipher text in Fig. 6, P denote a block of plain text, i denote a block index, and $D_K()$ denote block decryption with key K. Observing the decryption process in Fig. 6(b), we can see the following relationship.

$$P_i = C_{i-1} \text{ XOR } P_{i-1} \text{ XOR } D_K(C_i)$$

Clearly, obtaining each block of plain text P_i not only depends on its corresponding cipher text C_i, but also depends on its previous cipher text C_{i-1} and plain text P_{i-1}. To obtain P_i, we have to know P_{i-1}, and to obtain P_{i-1}, we have to know P_{i-2} and so on. As such, to decrypt any block of cipher text, we have to do the decryption from the beginning of a file. In contrast to CBC mode, we call this feature *block-decryption-dependent*.

Compared with recovering files encrypted with CBC mode, recovering files encrypted with PCBC mode is more difficult. We recover files encrypted with PCBC mode according to the following steps.

1. Estimate the physical disk data area where an encrypted file to be recovered could be allocated.
2. Find the first cluster of the file. Decrypt each cluster with an initialization vector and the remembered key K, and use individual cluster recognition approach [7,8] to find and decrypt the first cluster. Alternately, the first cluster can also be found from the directory entry table as shown in Fig. 1
3. Having the first cluster, we can find the second cluster. Decrypt each cluster with P and C of the last block of the first cluster and key K, and then use the individual cluster recognition approach to recognize the second cluster.
4. As such, we can find and decrypt the clusters 3, 4, ..., i.

Clearly, recovering files encrypted with PCBC mode is more difficult because failing to recover the ith cluster leads to failing to recover all clusters following the ith cluster.

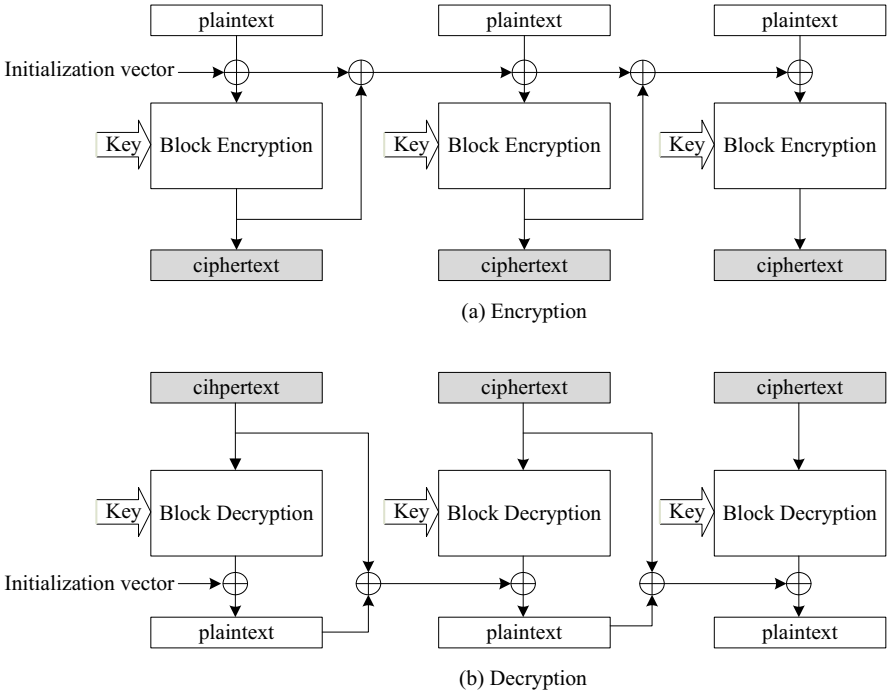

Fig. 6. Encryption and decryption processes of PCBC mode

4 Cipher Mode and Encryption Algorithm Recognition

In the previous section, we have presented the recovering approaches respectively for CBC and PCBC modes. The precondition is that we already know which mode was used to encrypt the file. In reality, however, the encryption mode is not known ahead of time. Furthermore, even if we know the cipher mode, we would still need to know what encryption algorithm is used inside a block encryption module. This section introduces an approach to recognize a cipher mode and an encryption algorithm.

Table 1. Classification of cipher modes

Feature	Cipher mode
block-decryption-dependent	PCBC, OFB
block-decryption-independent	CBC, ECB, CFB , CTS

In a cryptosystem, in addition to CBC and PCBC, there are other block cipher encryption modes. However, the number is limited. For example, Windows CryptoAPI [9] supports the cipher modes including, CBC, cipher feedback

(CFB), cipher text stealing (CTS), electronic codebook (ECB), output feedback (OFB). According to the decryption dependency, we classify these modes, as shown in Table 1. Since mode CBC, ECB, CFB, and CTS are in the same group, the approach of recovering files using mode ECB, CFB, and CTS is the same as that of recovering files using mode CBC, which has been presented in Section III-A. Similarly, the approach of recovering files using mode OFB is the same as that of recovering files using mode PCBC, which has been presented in Section III-B. Similar to cipher mode, the number of encryption algorithm for block cipher is also limited. Windows CryptoAPI [9] supports RC2, DES, and AES.

Algorithm 1: Cipher mode Recognition

```
Input: The first fragment of an encrypted file
Output: Cipher mode and encryption algorithm

    Step 1: Use RC2 as the encryption algorithm. Decrypt the
    first fragment respectively using mode CBC, ECB, CFB ,
    CTS, PCBC, and OFB, and save the corresponding decrypted
    plaintext fragments.

    Step 2: Use DES as the encryption algorithm. Decrypt the
    first fragment respectively using mode CBC, ECB, CFB ,
    CTS, PCBC, and OFB, and save the corresponding decrypted
    plaintext fragments.

    Step 3: Use AES as the encryption algorithm. Decrypt the
    first fragment respectively using mode CBC, ECB, CFB ,
    CTS, PCBC, and OFB, and save the corresponding decrypted
    plaintext fragments.

    Step 4: Recognize the first fragment from all plaintext
    fragments that are obtained from Step 1, 2, 3.

    Step 5: Output the cipher mode and the encryption algorithm
    corresponding to the recognized first fragment in Step 4.
```

We use an exhaustive algorithm to recognize the cipher mode and the encryption algorithm that are used to encrypt a to-be-recovered file. Algorithm **1** presents the steps of the recognition process. In Algorithm **1**, the beginning cluster number of the first fragment can be obtained from the directory entry table as shown in Fig. 1. If the used cipher mode and the encryption algorithm are included in Algorithm **1**, Step 5 must return correct results. It is worth noting that in Step 4 of Algorithm **1** we do not introduce a new file classification algorithm and we adopt the existing solutions [5].

5 Theoretical Analysis

In this section, we theoretically analyze the accuracy of recovering an entire encrypted file. For ease of presentation, we call this accuracy *Recovering Accuracy* (RA).

For recovering files with block-decryption-independent cipher mode, such as CBC and EBC, RA only depends on the recognition accuracy of a file because all contents (except the first block of a fragment as shown in Fig.5) of an encrypted file can be decrypted as plaintext. According to [6], based on the results, the recognition accuracy is variant for different file types. Table 2 [6] shows the results. Clearly, HTML file can be recognized with 100% accuracy and BMP file has the lowest accuracy. Nevertheless, as we present in Section III-A, the decrypted clusters that are not part of the to-be-recovered file become a random bit stream, which is favorable to classifying a decrypted file. Theoretically, RA should be higher than the results in Table 2.

Table 2. Recognition accuracy of different types of files [6]

Type	AVI	BMP	EXE	GIF	HTML	JPG	PDF
Accuracy	0.95	0.81	0.94	0.98	1.00	0.91	0.86

For recovering files with block-decryption-dependent cipher mode, such as PCBC and OFB, RA not only depends on the recognition accuracy of a file, but also on the number of clusters of an encrypted file. It is because recovering the ith cluster depends on whether the (i-1)th cluster can be recovered correctly. For ease of our analysis, we define some variables. Let k be the total number of clusters that a file has, p be the recognition accuracy, which is variant for different file types as shown in Table 2. Since the first cluster of a file can be found in a directory entry table, recognition accuracy on the first cluster is 100%. Therefore, we can derive RA related to k and p.

$$RA = p^{k-1}$$

Fig. 7 clearly shows the relationship between RA and p as increasing the number of clusters of a file (the size of a cluster is 4kb). As the number of clusters increases, RA decreases. On the other hand, the higher p is, the higher RA is. For some file types such as BMP file, since the recognition accuracy is relatively low ($p = 0.81$), RA becomes very low. However, for HTML file, since the recognition accuracy is relatively high ($p = 1$), RA is also high.

For cipher mode and encryption algorithm recognition, the recognition accuracy rate is the same as recognizing files with block-decryption-independent cipher mode, because only the first fragment of a file needs to be recognized. Also, this rate depends on the file type as shown in Table 2.

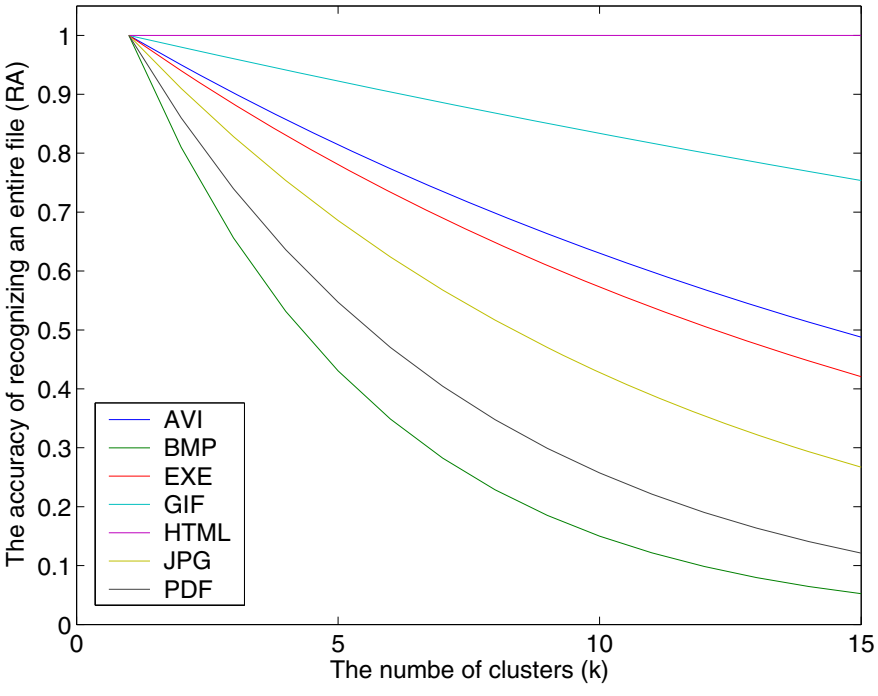

Fig. 7. Encryption and decryption processes of PCBC mode

6 Conclusions and Future Work

In this paper, we have identified the problem of recovering encrypted files, which depends on the encryption cipher mode and encryption algorithm. We have classified encryption cipher modes into two groups, block-decryption-dependant and block-decryption-independent. For each group, we have introduced a corresponding mechanism for file recovery. We have also proposed an algorithm to recognize the encryption cipher mode and the encryption algorithm with which a file is encrypted. Finally, we have theoretically analyzed the accuracy rate of recognizing an entire encrypted file.

We have reported a mechanism and an overall framework of recovering encrypted files. In the future, we will establish and implement an entire system for encrypted file recovery, especially, investigating the applicability of the proposed approaches on the various file/disk encryption solutions available currently, such as TrueCrypt [11], Encrypting File System (EFS) [12], which is a component of the New Technology File System (NTFS) file system on Windows for storing encrypted files. Further, in our system, we will include as many encryption algorithms as possible, including 3DES, AES-128, AES-192 and AES-256, and will also include stream cipher encryption mode. In addition, we will explore more promising recovery algorithms to accelerate the recovery speed.

Acknowledgements. We would like to thank the anonymous reviewers for their helpful comments. This work is partially supported by the grants from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. The MathWorks – MATLAB and Simulink for Technical Computing, http://www.mathworks.com/
2. MapleSoft – Mathematics, Mmodeling, and Simulation, http://www.maplesoft.com/
3. Pal, A., Memon, N.: The evolution of file carving. IEEE Signal Processing Magazine 26, 59–71 (2009)
4. McDaniel, M., Heydari, M.: Content based file type detection algorithms. In: 36th Annu. Hawaii Int. Conf. System Sciences (HICSS 2003), Washington, D.C (2003)
5. Wang, K., Stolfo, S.J.: Anomalous payload-based network intrusion detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203–222. Springer, Heidelberg (2004)
6. Veenman, C.J.: Statistical disk cluster classification for file carving. In: IEEE 3rd Int. Symp. Information Assurance and Security, pp. 393–398 (2007)
7. Karresand, M., Shahmehri, N.: File type identification of data fragments by their binary structure. In: IEEE Information Assurance Workshop, pp. 140–147 (2006)
8. Karresand, M., Shahmehri, N.: Oscar - file type identification of binary data in disk clusters and RAM pages. IFIP Security and Privacy in Dynamic Environments 201, 413–424 (2006)
9. Windows Crypto API, http://msdn.microsoft.com/enus/library/aa380255(VS.85).aspx
10. FAT – File Allocation Table, http://en.wikipedia.org/wiki/File_Allocation_Table
11. TrueCrypt – Free Open-source On-the-fly Encryption, http://www.truecrypt.org/
12. EFS – Encrypting File System, http://www.ntfs.com/ntfs-encrypted.htm