EVADING DISK INVESTIGATION AND FORENSICS USING A CLUSTER-BASED COVERT CHANNEL

HASSAN KHAN, MOBIN JAVED, FAUZAN MIRZA AND SYED ALI KHAYAM

WIRELESS AND SECURE NETWORKS (WISNET) RESEARCH LAB
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE (SEECS)
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY (NUST), ISLAMABAD, PAKISTAN

INTRODUCTION

Information hiding techniques have been proposed to evade disk forensics for investigation scenarios where the owner is compelled to release the decryption keys. A new plausible deniability approach to store sensitive information on a cluster-based file system is presented, which uses a covert channel to encode the sensitive information by modifying the fragmentation patterns of an existing file. The proposed covert channel does not require storage of any additional information and provides two-fold plausible deniability.

LIMITATIONS OF PROMINENT CONTEMPORARY APPROACHES

Steganographic FS [1] and StegFS [2] provides plausible deniability by writing random or encrypted contents on the disk.

However

→ Random and encrypted content on the disk can be easily detected.

→ Figure 1 illustrates this fact by showing that the entropy of encrypted content is significantly higher than unencrypted content.

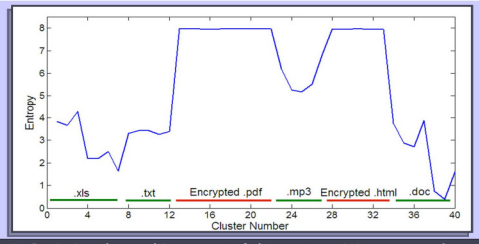


Figure1 : Deviation observed in entropy of clusters containing encrypted content

PROPOSED DATA HIDING ALGORITHM

Using the distance between fragmented clusters of a file to embed the hidden data.

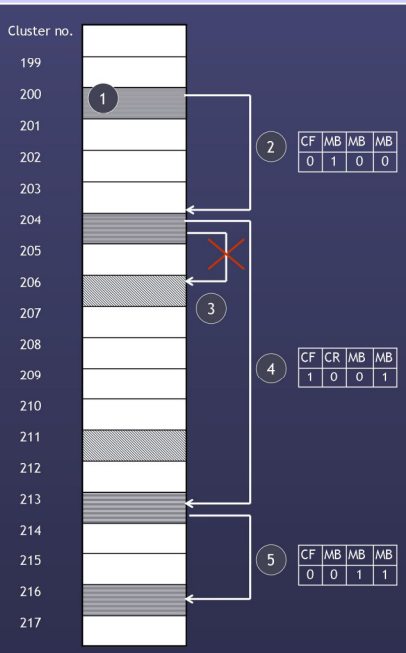


Figure 2: Hiding the binary message stream 10001011

Current coverfile contents | Existing coverfile contents

1. Store the first cluster contents at an arbitrary free cluster location.
2. Store the next cluster contents at: First cluster location + Data to be stored
   i.e. 200+ 4 = 204
3. Store the next cluster contents at: Previous cluster location +Data to be stored
   i.e. 204+ 2 = 206
   - → Data of a coverfile already exists at 206
   - → Set Collision Flag bit (CF bit) =1
   - → Try the first Collision Resolution Bit Value i.e. 0
4. Store the next cluster contents at: Previous cluster location +Modified Data to be stored
   i.e. 204+ 9 = 213
5. Store the next cluster contents at: Previous cluster location + Data to be stored
   i.e. 213+ 3 = 216

SALIENT FEATURES

Two-fold Plausible Deniability

→ The fragmentation is explained as phenomenon caused by multiple addition and deletion of files

→ If the fragmentation theory is not bought, the less confidential files are revealed only.

Hiding without actually storing

For hiding the data, the cluster distribution is only changed. That results in storing hidden data without actually writing on the disk.

PRELIMINARY EVALUATIONS

STORAGE CAPACITY

→Capacity is a decaying function of the number of allocated clusters.

→Maximum achievable capacity is 24000 hidden bits per 1000 clusters.

→Figure2 shows the cases where 0%,10%,25%, and 50% of occupied clusters belong to cover files.

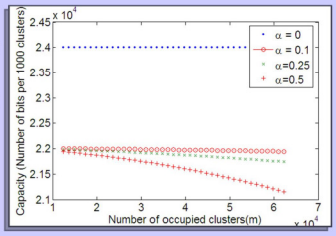
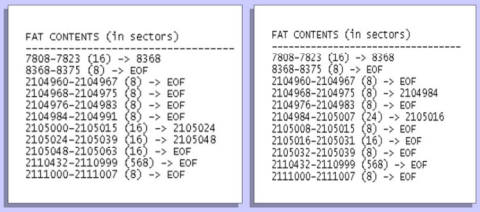


Figure 3 :Capacity plot for varying number of occupied clusters

SECURITY

Figure 4 shows it is always possible to fabricate an unintentional fragmentation (by addition and deletion of files of various sizes) which is very similar to an intentional Fragmentation (created in order to hide data).



```
FAT CONTENTS (in sectors)
-------------------------------------
7808-7823 (16) -> 8368
8368-8375 (8) -> EOF
2104960-2104967 (8) -> EOF
2104968-2104975 (8) -> EOF
2104976-2104983 (8) -> EOF
2104984-2104991 (8) -> EOF
2105000-2105015 (16) -> 2105024
2105024-2105039 (16) -> 2105048
2105048-2105063 (16) -> EOF
2110432-2110999 (568) -> EOF
2111000-2111007 (8) -> EOF
```

```
FAT CONTENTS (in sectors)
-------------------------------------
7808-7823 (16) -> 8368
8368-8375 (8) -> EOF
2104960-2104967 (8) -> EOF
2104968-2104975 (8) -> 2104984
2104976-2104983 (8) -> EOF
2104984-2105007 (24) -> 2105016
2105008-2105015 (8) -> EOF
2105016-2105031 (16) -> EOF
2105032-2105039 (8) -> EOF
2110432-2110999 (568) -> EOF
2111000-2111007 (8) -> EOF
```

Figure 4 : (a) Intentional Fragmentation (b) Unintentional Fragmentation

REFERENCES

[1] R. Anderson, R. Needham, and A. Shamir, "The Steganographic Filesystem," in Proc. IH98.

[2] A. McDonald and M.G. Kuhn, "Stegfs: A Steganographic Filesystem for Linux," in Proc. IH99.