

Structural Artifacts of Secure Deletion with Full Disk Encryption

Andrew Brusso

Michigan Technological University

April 18, 2019

Overview

1 Background

- Secure Deletion
- Full Disk Encryption

2 The Problem

- Motivation
- Approach

3 Results

What is Secure Deletion?

- When deleting a file on a typical file system, deletion only changes metadata to make it look like the file is gone.
- Secure deletion uses either encryption or overwriting to more completely delete the data itself, instead of just changing the metadata.

Example of Non-secure Deletion

- 1 Freshly formatted FAT filesystem
- 2 3 files added to filesystem
- 3 File 2 deleted from drive

```
$START_SNAPSHOT
Page 0| ě<MSDOS5.0
Page 1| øÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6|
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14|
Page 15|
Page 16|
Page 17|
Page 18|
Page 19|
```

```
$START_SNAPSHOT
Page 0| ě<MSDOS5.0
Page 1| øÿÿÿÿÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6| FILE1      | |aëNN
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14| file1 contentsfile1
Page 15|
Page 16| file2 contentsfile2
Page 17|
Page 18| file3 contentsfile3
Page 19|
```

```
$START_SNAPSHOT
Page 0| ě<MSDOS5.0
Page 1| øÿÿÿÿÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6| FILE1      | |aëNN
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14| file1 contentsfile1
Page 15|
Page 16| file2 contentsfile2
Page 17|
Page 18| file3 contentsfile3
Page 19|
```

Example of Secure Deletion

- 1 Freshly formatted FAT filesystem
- 2 3 files added to filesystem
- 3 File 2 *securely* deleted with windows sdelete utility (overwrite with zeroes)

```
$START_SNAPSHOT
Page 0| ě<MSDOS5.0
Page 1| øÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6|
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14|
Page 15|
Page 16|
Page 17|
Page 18|
Page 19|
```

```
$START_SNAPSHOT
Page 0| ě<MSDOS5.0
Page 1| øÿÿÿÿÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6| FILE1      [aëNN
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14| file1 contentsfile1
Page 15|
Page 16| file2 contentsfile2
Page 17|
Page 18| file3 contentsfile3
Page 19|
```

```
$START_SNAPSHOT
Page 0| ě<MSDOS5.0
Page 1| øÿÿÿ
Page 2|
Page 3|
Page 4|
Page 5|
Page 6| FILE1      [aëNN
Page 7|
Page 8|
Page 9|
Page 10|
Page 11|
Page 12|
Page 13|
Page 14| file1 contentsfile1
Page 15|
Page 16|
Page 17|
Page 18| file3 contentsfile3
Page 19|
```

Secure Deletion isn't *Truly* Secure

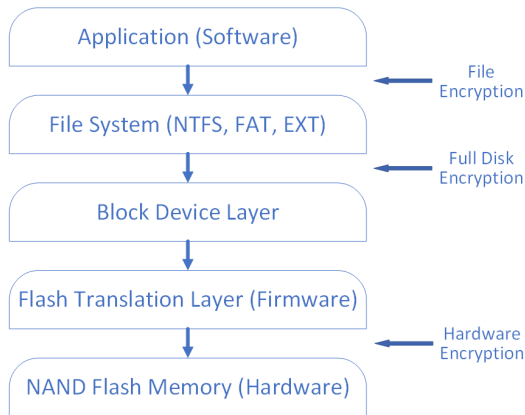
For a deletion scheme to be **Truly Secure**, it needs two properties [1]

- ① Data is sanitized so that attacker cannot access it
- ② Structural artifacts are removed so that adversary cannot *infer any* sensitive information about the deleted data

Flash memory's log structured writing creates structural artifacts with conventional secure deletion.

What is Full Disk Encryption?

- Full Disk Encryption uses a block cypher (often AES) at the block device layer. Everything is encrypted, including the File System.
- A pre-boot sequence prompts for a password. This “unlocks” the device, and then all data is encrypted at the **sector** level. [2][3][6]



The Problem

Does conventional secure deletion in Full Disk Encryption meet the requirements to be **Truly Secure**?

- A secure deletion will sanitize the data, just like it sanitizes in the case of plaintext.
- Is there any leakage or structural artifacts that appear when sanitizing inside a Full Disk Encryption scheme?

Why does this matter?

- Current secure deletion schemes that address structural artifacts for Flash Memory either require specialized file systems (YAFFS) [4][5], or implementations at the Flash Translation Layer[1], which only works with supported hardware.
- If FDE meets criteria for truly secure deletion, it could offer an alternative approach for users with existing systems to adopt a truly secure deletion scheme at minimal cost.

Adversarial Model

- The attacker doesn't have the encryption key, otherwise they can just decrypt and it is the same as the plaintext case.
- Attacker is able to take **multiple** snapshots of the raw flash. If you have only one snapshot, it is just random data.
- Attacker needs to be able to take a snapshot prior to data being added, after data is added, and after it is deleted, otherwise can't discern data changed after being added.

My Approach

- ① Created a way to take snapshots of the raw flash on demand
- ② Identified three candidate Full Disk Encryption programs: Veracrypt, CipherShield, and Symantec PGP Whole Disk Encryption
- ③ Encrypted flash device with each FDE
- ④ Performed a sequence of actions and took snapshots between actions
- ⑤ Observed changes between snapshots, and looked for signs of structural artifacts or leakages.

Add 3 files

Secure delete file 2

Before

After

```

66 e637b82bca2e1b5b728835cf2a8cd1f
67 7f2a2d6736354db547bb3a4f51331
68 f3df9a38f9e385cfbc5182ebdbd1f1f
69 bf2cc3859fa342f782d23594a0c162
70 23d0d6c02c35ac3f14c0263b6310a8
71 9168deb6ba7cef43e451d3d91c3151
72 4243df97de58964944c0879519d0f
73 7129dc74306343ecf84d0f72c86e1f
74 d05cf4d4ed8ee02c27e8dd797141f1
75 4fef59c86938487f0a3bdf69b6a9b
76 4d89c5b6e0fd16afb8bc79ebb9691d
77 d440f52ef591ca136abfe19db2982e
78 abdec82cd28cb5b52d05449246fda1f
79
80 a3ce622ef9589cb2a1b12c563b8fa
81 e28df0dd68f58dc92a11256d04c777f
82 96c8ad8cf75f5fed93f88dbef1328b
83 a4faed67a3d3a37fb6436a3f9df818f
84 294a6bc1e43f79884d7053c37cf594
85 ed6abdd6d276da5b76c1ae09577d3c
86 2e5b8837a561d21936578ca586cd1cc
87 3acde271911cf7156463f55d5c4f6c
88 22db997eb4115d5789c72d6fa92abc
89 24fa5138e418c87046f5c9f182c18f
90 f66b9fedf72b0cdce576b652a2af1f
91 8679368ff07c92f265c2d2ae72fbc
92 46b4ae6a3715e7a118c5f799dc9d0f
93 a7637fec5b28de64e92321f2dc3a5f
94 198789f8e8374d79d71fddad8b6e3c
95 116736572d7b1f976d1f18e5a5e5f
96 b54762f745419aac5ec14f58b7688c1
97 f95f8afae26729c3baab767b2941582
98 8729ef40994c7674a1184adcfb15967f
99 32c6147c18893346d4e9c8e8d8f9f
100 b04dc7ec62b6e69c9f996ba4a11455
101 cdd6bc375c11baeac6becac094c74f
102 9ad626757a3674137338f73c13e74d
103 d1ec215474a7752483aa7f8a12b044f
104 f5b3fa8b254c257c5b05d19e989f86
105 e0841f77cb1a5af7d1222a9c5a568f
106 eee2f923213a3d411c4e4e9da735f
107 906b1f765fe9592ad6220487cadedf
108 271999e04e68444d476909c3931622
109 c72ab258078ebbf55384543380ef8f
110 8bd776dc553f39d9347c4ec1eccc7f
111 fb2b71a8c7d8b9a23c63540f84d5f
112 bcd7f8089c87d8ba9a23c63540f84d5f
113 50a2e7dkc61a5b3ebbf0d6f153a794f
114 e1ca4263ba6c5b65d73cb2b793147c
115 c36eb630c5a0e654f785196a9e2e1f
116 771371e5932c541cc423f6c87875f
117 7398c9c999d19a658fa363f153a018f
118 6ccdd93167371f7d5f4818d377f0d

```

Before

After

```

66 e637b82bca2e1b5b728835cf2a8cd1f
67 29a1ada1f98728754ff2e20a51e1f
68 f3df9a38f9e385cfbc5182ebdbd1f1f
69 bf2cc3859fa342f782d23594a0c162
70 23d0d6c02c35ac3f14c0263b6310a8
71 9168deb6ba7cef43e451d3d91c3151
72 40495cc0446a69f441cd3f313de9f
73 7129dc74306343ecf84d0f72c86e1f
74 d05cf4d4ed8ee02c27e8dd797141f1
75 4fef59c86938487f0a3bdf69b6a9b
76 4d89c5b6e0fd16afb8bc79ebb9691d
77 d440f52ef591ca136abfe19db2982e
78 abdec82cd28cb5b52d05449246fda1f
79
80 f1a1a61b9954cfb6ca81cd4cbdc5a1f
81 1ede633f4541e7765a0e2816475db
82 559b2bba79a44f1e9c33aaf6fcdcf
83 343579a68ede5dc3d346b1f280886c
84 70115b26c1c45f633dd89d4a51laa
85 488ed8656f12
86 cfad28b5261b236c08981dd472c
87 1173df86b39022e8aabd3bac5247b
88 c98d5fca77aacd7f1f4bf366828b
89 875fd9c543cad52964342a30bba
90 3918f842af1cdcb7f9a7389d58f0c
91 8679368ff07c92f265c2d2ae72fbc
92 ca8cf688bb5f
93 4c9da9d73d91248690adc6389f3cc
94 ba43667ea36f137e56d216cc359211
95 d592d8fdeaf0e3644272d3739d6c5
96 7e74ear52025185f7a12b99169a71f
97 e2bb
98 4f4dc6384874f5d78333ec72e2c27b
99 ac514f6f711692b6b44fd3d127f0
100 22d67ba0fe2027574b387c18567f
101 687fab25e2a085a3cb075f1ca3266d
102 b95f2a1510b84c274584216f9a51f
103 d1ec215474a7752483aa7f8a12b044f
104 488c5ba84a769035deb51e6f951f
105 61e669ebff94e714c3fd9b2db0482
106 5304f5ac977dcbec1ee3578bdad
107 f6688f1fc7dc1962a7c4f91f7d1e4t
108 6789ba90d2a44f8dc062c63e2f1b1
109 5da591b1c535fedeccce9fa266ea
110 bc67b7a85117c2a79c76ad517af6e
111 44fa8e32e12e5f23858a4633adbd2
112 3f4c36984d838015dc2e375107f
113 10be2ec5d53488151898444ad29c
114 edac62d1f7c20e155b13c90ca42
115 edac62d1f7c20e155b13c90ca42
116 c36eb630c5a0e654f785196a9e2e1f
117 771371e5932c541cc423f6c87875f
118 7398c9c999d19a658fa363f153a018f
119 6ccdd93167371f7d5f4818d377f0d

```

```

66 e637b82bca2e1b5b728835cf2a8cd1f
67 29a1ada1f98728754ff2e20a51e1f
68 f3df9a38f9e385cfbc5182ebdbd1f1f
69 bf2cc3859fa342f782d23594a0c162
70 23d0d6c02c35ac3f14c0263b6310a8
71 9168deb6ba7cef43e451d3d91c3151
72 40495cc0446a69f441cd3f313de9f
73 7129dc74306343ecf84d0f72c86e1f
74 d05cf4d4ed8ee02c27e8dd797141f1
75 4fef59c86938487f0a3bdf69b6a9b
76 4d89c5b6e0fd16afb8bc79ebb9691d
77 d440f52ef591ca136abfe19db2982e
78 abdec82cd28cb5b52d05449246fda1f
79
80 f1a1a61b9954cfb6ca81cd4cbdc5a1f
81 1ede633f4541e7765a0e2816475db
82 559b2bba79a44f1e9c33aaf6fcdcf
83 343579a68ede5dc3d346b1f280886c
84 70115b26c1c45f633dd89d4a51laa
85 488ed8656f12
86 cfad28b5261b236c08981dd472c
87 1173df86b39022e8aabd3bac5247b
88 c98d5fca77aacd7f1f4bf366828b
89 875fd9c543cad52964342a30bba
90 3918f842af1cdcb7f9a7389d58f0c
91 8679368ff07c92f265c2d2ae72fbc
92 ca8cf688bb5f
93 4c9da9d73d91248690adc6389f3cc
94 ba43667ea36f137e56d216cc359211
95 d592d8fdeaf0e3644272d3739d6c5
96 7e74ear52025185f7a12b99169a71f
97 e2bb
98 4f4dc6384874f5d78333ec72e2c27b
99 ac514f6f711692b6b44fd3d127f0
100 22d67ba0fe2027574b387c18567f
101 687fab25e2a085a3cb075f1ca3266d
102 b95f2a1510b84c274584216f9a51f
103 d1ec215474a7752483aa7f8a12b044f
104 488c5ba84a769035deb51e6f951f
105 61e669ebff94e714c3fd9b2db0482
106 5304f5ac977dcbec1ee3578bdad
107 f6688f1fc7dc1962a7c4f91f7d1e4t
108 6789ba90d2a44f8dc062c63e2f1b1
109 5da591b1c535fedeccce9fa266ea
110 bc67b7a85117c2a79c76ad517af6e
111 44fa8e32e12e5f23858a4633adbd2
112 3f4c36984d838015dc2e375107f
113 10be2ec5d53488151898444ad29c
114 edac62d1f7c20e155b13c90ca42
115 edac62d1f7c20e155b13c90ca42
116 c36eb630c5a0e654f785196a9e2e1f
117 771371e5932c541cc423f6c87875f
118 7398c9c999d19a658fa363f153a018f
119 6ccdd93167371f7d5f4818d377f0d

```

Results

- File system state with deletion performed vs. if it had never been performed is different, so scheme is not truly secure.
- Edits can be discerned from deletes if file size $>$ size of a sector, and edits don't span every page.
- Not secure deleting at all might be **better**? Secure deletion causes changes to the data itself between snapshots, so artifacts are immediately apparent
- There are structural artifacts, but are they actually useful? Without the encryption keys you may be able to say a deletion occurred, but you don't have the file context like you do in the plaintext case. Can say some file was deleted, but anything else?

Potential Future Work

- Scripting interactions with dev board for reliability/repeatability. E.g. script generates test files, performs experiment actions, and then automatically takes snapshots.
- Scripted analysis with the snapshots. E.g. script that runs diffs between snapshots and tries to classify changes as add, delete and edit. Analysis of adds/deletes/edits in aggregate might help identify leakages?
- Experiment with different file systems. FAT is very simple, NTFS or EXT potentially introduces new difficulties.
- Windows 10 compatibility, test with Bitlocker, which utilizes TPM and AES CBC.

References

- [1] Bo Chen et al. “Sanitizing Data is Not Enough!: Towards Sanitizing Structural Artifacts in Flash Media”. In: *Proceedings of the 32Nd Annual Conference on Computer Security Applications. ACSAC '16*. Los Angeles, California, USA: ACM, 2016, pp. 496–507.
- [2] Niels Ferguson. *AES-CBC + Elephant diffuser: A disk encryption algorithm for Windows Vista*. 2006.
- [3] *How Whole Disk Encryption Works*. Symantec Corporation, Nov. 2010.
- [4] Jaeheung Lee et al. “An Efficient Secure Deletion Scheme for Flash File Systems”. In: *J. Inf. Sci. Eng.* 26 (2010), pp. 27–38.
- [5] Joel Reardon, Srdjan Capkun, and David A. Basin. “Data Node Encrypted File System: Efficient Secure Deletion for Flash Memory”. In: *USENIX Security Symposium*. 2012.
- [6] *Veracrypt Technical Details*. Veracrypt. URL: <https://www.veracrypt.fr/en/Technical%20Details.html>.