Forensic Decryption of FAT BitLocker Volumes

P. Shabana Subair, C. Balan(✉), S. Dija, and K.L. Thomas

Centre for Development of Advanced Computing, PO Box 6520,
Vellayambalam, Thiruvananthapuram 695033, Kerala, India
{shabana, cbalan, dija, thomaskldija}@cdac.in

Abstract. New versions of Windows come equipped with mechanisms, such as EFS and BitLocker, which are capable of encrypting data to an industrial standard on a Personal Computer. This creates problems if the computer in question contains electronic evidence. BitLocker, for instance, provides a secure way for an individual to hide the contents of their entire disk, but as with most technologies, there are bound to be weaknesses and threats to the security of the encrypted data. It is conceivable that this technology, while appearing robust and secure, may contain flaws, which would jeopardize the integrity of the whole system. As more people encrypt their hard drives, it will become harder and harder for forensic investigators to recover data from Personal Computers. This paper documents the Bitlocker Drive Encryption System (version 2) in Windows 7. In particular it describes how to forensically decrypt and load a FAT disk or image which is bitlocked, if the keys are provided.

Keywords: Bitlocker To Go · Bitlocker keys · Full volume encryption key · Volume master key · AES-CCM · Elephant diffuser · AES-CBC

1 Introduction

Volumes encrypted with BitLocker will have a different signature than the standard NTFS header. Instead, they have in their volume header (first sector): 2D 46 56 45 2D 46 53 2D or, in ASCII, -FVE-FS-.

These volumes can be identified by the BitLocker GUID/UUID: 4967d63b-2e29-4ad8-8399-f6a339e3d00.

The actual data on the encrypted volume is protected with either 128-bit or 256-bit AES or optionally diffused using an algorithm called Elephant. The key used to do the encryption, the Full Volume Encryption Key (FVEK) and/or TWEAK key, is stored in the BitLocker metadata on the protected volume. The FVEK and/or TWEAK keys are encrypted using another key, namely the Volume Master Key (VMK). Several copies of the VMK are also stored in the metadata. Each copy of the VMK is encrypted using another key; also know as key-protector key. Some of the key-protectors are:

- TPM (Trusted Platform Module)
- Smart card
- recovery password

P. Gladyshev et al. (Eds.): ICDF2C 2013, LNICST 132, pp. 17–29, 2014.
DOI: 10.1007/978-3-319-14289-0_2

- start-up key
- clear key; this key-protector provides no protection
- user password

BitLocker has support for partial encrypted volumes.

1.1 BitLocker To Go

BitLocker To Go is a full-disk encryption protection technology for removable storage devices. Though it is based on BitLocker technology, BitLocker To Go significantly enhances the technical capabilities of BitLocker. For example, it is compatible with all FAT (FAT32, exFAT, etc.) file systems in addition to NTFS, dramatically increasing its compatibility with existing devices.

Volumes encrypted with BitLocker To Go will have a hybrid encrypted volume, meaning that part of the volume is unencrypted and contains applications to unlock the volume and the other part of the volume is encrypted. The "discovery drive" volume contains BitLocker To Go Reader to read from encrypted volumes on versions of Microsoft Windows without BitLocker support.

BitLocker To Go is designed primarily for enterprises, where there is serious risk of a user bringing an unprotected storage device into the environment, copying important corporate information (inadvertently or not) to it, and then losing the device outside of the workplace. USB memory keys, in particular, are small and convenient, and quite popular, but they're also easily lost. With BitLocker To Go enabled on the device, one can help protect sensitive corporate–or, for that matter, personal–data in the event of loss or theft.

BitLocker To Go works completely independently of BitLocker, so you do not need to enable BitLocker on the PC, or utilize any TPM hardware, in order to use BitLocker To Go. In use, however, it is similar to BitLocker, and can also be enabled via a simple right-click menu choice.

This paper contains the details necessary to access Bitlocker protected FAT volumes. It describes the Bitlocker recovery information like the BitLocker keys, the encryption methods, the details of volume header, the metadata block and about the metadata header and metadata entries. Finally this paper presents the steps to unlock a BitLocker FAT32 volume.

2 Bitlocker Recovery Information

2.1 Bitlocker Keys

The BitLocker key management system uses a series of keys to protect the data at rest. This section describes the various keys that are used in the BitLocker encryption process as they have been documented by Microsoft.

2.1.1 Full Volume Encryption Key (FVEK)

The key used to protect the data i.e. the sector data is the Full Volume Encryption Key. It is stored on the protected volume and is stored encrypted. To prevent unauthorized

access the FVEK is encrypted with the Volume Master Key (VMK). The size of the FVEK is dependent on the encryption method used i.e. FVEK is 128-bit of size for AES 128-bit and FVEK is 256-bit for AES 256-bit.

2.1.2 Volume Master Key (VMK)

The key used to encrypt the FVEK is the Volume Master Key (VMK). It is also stored on the protected volume. The VMK is 256-bit. In fact several copies of the VMK are stored on the protected volume. Each copy of the VMK is encrypted using a different key such as the recovery key, external key, or the TPM. If the volume is bitlocked using both external key as well as the recovery password, then there will be two metadata entries for VMK where each metadata entry stores the VMK encrypted with recovery key and the external key respectively. When decrypted, both the VMK will be the same. If the VMK differ then it means that the decryption has failed.

It is also possible that the VMK is stored unencrypted which is referred to as clear key.

2.1.3 TWEAK Key

The TWEAK is part of the FVEK stored encrypted with the Volume Master Key (VMK). The size of the TWEAK key depends on the encryption method used. The key is 128-bit for AES 128-bit and the key is 256-bit for AES 256-bit.

The TWEAK key is present only when the Elephant Diffuser is enabled. The TWEAK key is stored in the metadata entry that holds the FVEK which is always 512-bit. The first 256-bits are reserved for the FVEK and the other 256-bits are reserved for the TWEAK key. Only 128-bit of the 256-bits are used when the encryption method is AES 128-bit i.e. when the Elephant Diffuser is disabled.

2.1.4 Recovery Key

BitLocker stores a recovery (or numerical) password to unlock the VMK. This recovery password is stored in a {%GUID %}.txt file.

Example recovery password: 471207-278498-422125-177177-561902-537405-468006-693451.

The recovery password is valid only if it consists of 48 digits where every 6 numbers are grouped into a block thus consisting of 8 blocks. Here each block should be divisible by 11 yielding a remainder 0. The result of a division by 11 of a block is a 16-bit value. The individual 16-bit values make up a 128-bit key.

2.1.5 External Key

The External key is stored in a file named {%GUID %} .BEK. The GUID in the filename equals the key identifier in the BitLocker metadata entry. The BEK file contains the external key identifier and a 32 byte external key.

The different keys allow different mechanisms to be used to access the stored data. Each access mechanism can be used to decrypt a copy of the VMK which in turn is used to decrypt the FVEK which in turn is used to decrypt the protected data.

2.2 Encryption Methods

BitLocker uses two encryption methods to encrypt the data. First it uses the AES-CBC with or without Elephant Diffuser to encrypt the sector data i.e. the main data. Second it uses the AES-CCM to encrypt the keys like the VMK and FVEK.

2.3 Unencrypted Sector(s)

In BitLocker the sectors that are stored as unencrypted sectors are

- The unencrypted volume header
- The BitLocker metadata

Both BitLocker Windows 7 and To Go store an encrypted version of the first sectors in a specific location. This location is found in the FVE metadata block header. It is the location where the original boot sector starts.

At this point we shall describe the important offsets that are of interest to a forensic examiner in the volume header, metadata block header, metadata header and the content of the metadata entries. Specifically the paper explains how to decrypt a bit-locked drive.

We'll begin with the Bitlocker Volume Header, and then explain the details of the metadata block, the metadata header and the metadata entries. Finally the last section explains the steps to decrypt the sectors of the bitlocked volume and replace them so that the original volume is recovered.

2.4 Volume Header

The BitLocker Windows To Go volume header for a FAT volume is similar to FAT32 boot sector. Let us refer this duplicate boot sector as the volume header from now on. This volume header is 512 bytes of size. If a FAT volume (i.e. it is either FAT12 or FAT16 or FAT32 or exFAT) is bitlocked, the volume header is same as that of the FAT32 boot sector. The important information that the volume header consists of is the offsets of the three FVE metadata blocks [4].

From the volume header we get the offsets of the three FVE metadata blocks which contain the offset of the original boot sector which is stored encrypted in some location.

2.5 FVE Metadata Block

A Bitlocker protected volume contains three identical metadata blocks for redundancy. Even though the first metadata block gets damaged, the second and third metadata block can be used to get the original boot sector offset. As shown in the Fig. 1, the volume header contains the offsets of the three metadata blocks.

To find metadata blocks on a damaged volume, the examiner can search the volume for the metadata signature -FVE-FS-. Because each metadata block can only begin at offsets that are a multiple of the bytes per sector and sectors per cluster, the examiner could speed up the search by only searching for the string at these offsets. To be safe,

424	16		**BitLocker identifier** **contains a GUID**
440	8		**FVE metadata block 1 offset** **Contains an offset relative to the start of the volume**
448	8		**FVE metadata block 2 offset** **Contains an offset relative to the start of the volume**
456	8		**FVE metadata block 3 offset** **Contains an offset relative to the start of the volume**

Fig. 1. The data showing the metadata block offsets in the Volume header [4]

the examiner should assume the smallest legal values and thus search for the BitLocker signature at multiples of 512 bytes.

Each FVE metadata block consists of:

1. A FVE metadata block header
2. A FVE metadata header.
3. An array of FVE metadata entries

2.6 FVE Metadata Block Header

The FVE metadata block header consists of the signature "-FVE-FS". The metadata block is valid only if the signature is present. The size field indicates the size of the metadata block:

Offset	Size	Value	Description
16	8		Encrypted volume size Contains the number of bytes
24	4		
28	4		Number of volume header sectors Contains the number of sectors
32	8		FVE metadata block 1 offset Contains an offset relative to the start of the volume
40	8		FVE metadata block 2 offset Contains an offset relative to the start of the volume
48	8		FVE metadata block 3 offset Contains an offset relative to the start of the volume
56	8		**Volume header offset** **Contains an offset relative to the start of the volume**

Fig. 2. Structure of FVE metadata block header [4]

Each Bitlocker metadata block begins with a variation length header followed by a variable number of entries. The FVE metadata block header contains the offset of the original boot sector. When decrypting, BitLocker will decrypt from the back to the front. The encrypted volume size at offset 16 contains the number of bytes of the volume that are encrypted or need to be decrypted (Fig. 2).

2.7 FVE Metadata Header

The FVE metadata header is 48 bytes. There are several pieces of forensically valuable data in the metadata header. First, the volume's Global unique Identifier (GUID) is stored at offset 16. This GUID should be included on any access device that unlocks this device such as USB sticks. Examiners can search for this GUID on USB devices to find possible Bitlocker access devices. The date and time Bitlocker was enabled is recorded at the offset 40. Finally the next counter value to be used for key encryption nonce is stored at the offset 32. As mentioned earlier, this could be useful in determining how many access devices have been created for a volume. Also, the encryption method at offset 36 describes the type of encryption that has been used to encrypt the volume.

2.8 FVE Metadata Entry

There is an array of FVE metadata entries and each metadata entry is variable size and consists of entry size, entry type and so on. If the volume is bitlocked using both recovery password and external key, then there will be metadata entries for both the recovery key and the external key. The Fig. 3 shows the FVE metadata entry structure as taken from [4].

Offset	Size	Value	Description
0	2		Entry size
2	2		Entry type
4	2		Value type
6	2	1	Version
8	…		Data

Fig. 3. FVE metadata entry values

2.9 FVE Metadata Entry Types

Each value in the entry type filed indicates which type of key the metadata entry stores.

2.10 FVE Metadata Value Types

The metadata value types indicate whether the key is erased, or whether it is a stretch key, or volume master key.

The Bitlocker metadata header is followed by a series of metadata entries. These entries contain the encrypted FVEK and several copies of the VMK. Each copy of the VMK is encrypted with a different key. If for example a volume is bitlocked using only the recovery key then the VMK will be encrypted using the recovery key whereas if the volume is bitlocked using both recovery key and external key then the VMK will be encrypted with the two keys and stored separately.

Each metadata entry consists of data concerning where the key in question is stored and has at least two encrypted key protectors. The first key protector structure contains a copy of the VMK encrypted with the key and the second key protector structure contains a copy of the key encrypted using the VMK. The timestamps are identical in both the key protectors. The same is applicable for VMK and FVEK. The first key protector contains a copy of the FVEK encrypted using the VMK and the second key protector contains a copy of the VMK encrypted using FVEK. Now as all the details are explained, the next section deals with the steps of decrypting and loading a FAT Bitlocker volume.

3 Loading a Bitlocked Volume

In this section we describe the steps needed to decrypt and load a FAT bitlocked volume. Here an evidence file that has two partitions has been taken where one is FAT32 volume that is bitlocked. The.txt file having the recovery password and a.bek file having the external key which is generated are stored on some external drive. Both these files are used for the recovery process and using these keys, the VMK and FVEK are derived. The process is same for all the other FAT file systems.

3.1 Derivation of Keys

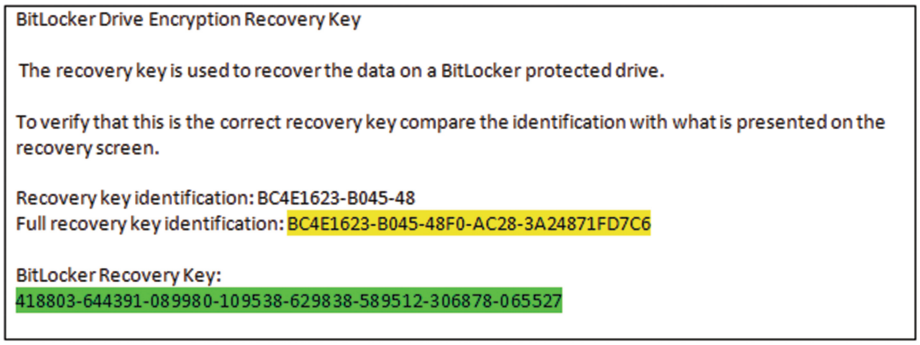
Here the first sector of the bitlocked volume does not contain the original boot sector. Instead it contains the bitlocked volume header which is similar to the FAT32 boot sector. Though it is similar some values are changed and provide other valuable information as shown in the Table 1. The important information that the volume header contains is the offsets of the three metadata blocks. So by getting the offset of the first metadata block, the metadata block header is read and the signature –FVE-FS is checked so that a valid metadata block is read.

The metadata block consists of two important information. One is the original boot sector starting sector information and the second is the encrypted volume size. There are important while recovering the bitlocked volume.

Next the metadata header and metadata entries are read and the key protector structures are stored. The key protector structures contain the encrypted keys which have to be decrypted, the protection type, GUID and the last modification time.

3.1.1 Decryption of the VMK Using Recovery Password

The recovery password is the 48-digit key that is taken from the.txt file. The Fig. 4 shows the text file containing the 48-digit recovery key and the GUID highlighted.


BitLocker Drive Encryption Recovery Key

The recovery key is used to recover the data on a BitLocker protected drive.

To verify that this is the correct recovery key compare the identification with what is presented on the recovery screen.

Recovery key identification: BC4E1623-B045-48
Full recovery key identification: BC4E1623-B045-48F0-AC28-3A24871FD7C6

BitLocker Recovery Key:
418803-644391-089980-109538-629838-589512-306878-065527

Fig. 4. The .txt file showing the GUID and the 48 digit recovery key

As described in Sect. 2, the individual 16-bit values make up a 128-bit key. The corresponding recovery key is calculated using the following approach:

```
Initialize a structure consisting of:
uint8_t last_sha256 [32 ];
uint8_t initial_sha256 [32 ];
uint8_t salt[ 16 ];
uint64_t count;
```

Initialize both the last SHA256 and the count to 0. Calculate the SHA256 of the 128-bit key and update the initial SHA256 value. The salt is stored on disk in the stretch key which is stored in the recovery key protected Volume Master Key (VMK). Loop for 1048576 (0 × 100000) times:

- calculate the SHA256 of the structure and update the last SHA256 value
- increment the count by 1

The last SHA256 value contains the 256-bit key which is recovery key that can unlock the recovery key protected Volume Master Key (VMK).

3.1.2 Decryption of the VMK Using Startup Key

The generated .bek file is used along with the metadata entry. Both the bek file and the metadata entry contain the same Globally Unique Identifier (GUID). This allows for the correct matching of the BEK with the metadata entry and also for checking the validity of the BEK file.

The external key from the.bek file is extracted and the data from the key protector structure is decrypted using this external key which unlocks the VMK. The Fig. 5 shows the external key highlighted in the BEK file.

3.1.3 Decryption of FVEK

The encrypted FVEK is stored in the key protector structure of the metadata entry which is highlighted as shown in the Fig. 6.

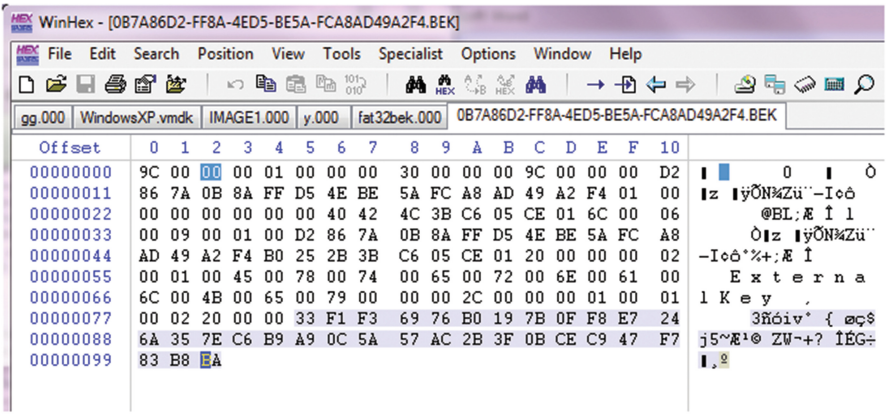


Fig. 5. The BEK file of the Bitlocker volume

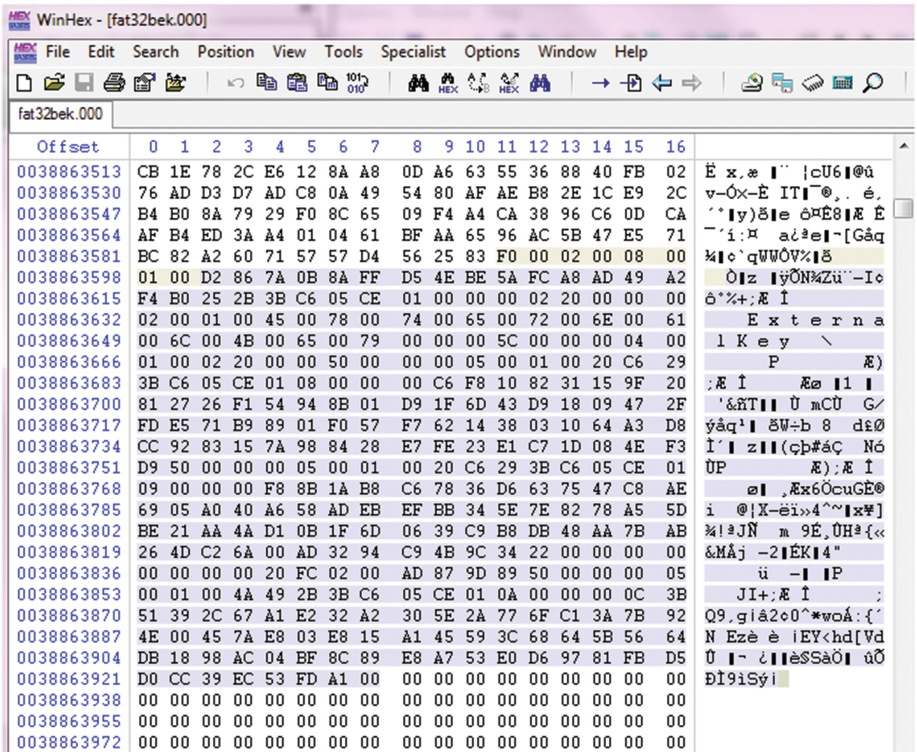


Fig. 6. The metadata entry showing the external key

The encrypted data is decrypted using the VMK and the type of algorithm as specified in the metadata entry. Finally the FVEK has been derived which is used to decrypt the data at rest.

3.2 Replacement of the Original Sectors

Though the bitlocked volume contains a volume header it is not the correct boot sector. The original boot sector is stored encrypted somewhere in the volume. So we have to find the original boot sector, decrypt it and then replace the volume header with the original boot sector. In the previous section it has been explained that the metadata block header contains the original boot sector starting offset. So by getting the offset and by having the sector, the steps for decrypting the sector are explained.

Here the data has to be decrypted sector by sector. The steps in decrypting a sector of data are:

1: Decrypt the data with FVEK in AES-CBC mode
2: Run Diffuser B in decryption direction 3 times
3: Run Diffuser A in decryption direction 5 times
4: Calculate Sector key from the TWEAK key
5: XOR data obtained in step 3 with Sector Key
6: Plaintext

3.2.1 Correct Intake of Sector Number

For successful decryption, one of the important things that should be considered is the sector number. In the first step, we have to calculate the Initialization Vector. To calculate the Initialization Vector, the sector number is used. The sector number is very important because if we give the wrong sector number, the data will not be decrypted correctly and we get the wrong data. So the sector number is of very much importance. The decryption might fail due to the wrong intake of the sector number.

For the evidence file explained above, the original boot sector starts at the sector number 1404 and the partition starts at 63. So if we take the sector number of the boot sector as 1404 and decrypt it, we do not get a valid boot sector but some junk data. So here the sector number should be taken as:

Sector number of the boot sector to be taken = original sector number of the boot sector – partition start sector.

Hence in the above case in order to get the correct boot sector we have to take the sector number 1341(1404-63) instead of 1404. In this way the valid boot sector is placed at the front. The remaining sectors are also decrypted in the same way and rearranged accordingly.

The Figs. 7 and 8 show the encrypted boot sector and the decrypted boot sector of the bitlocked volume.

3.2.2 Decryption of Sectors After Encrypted Volume Size is Reached

The next important issue to be considered is the decryption of the sectors after the encrypted volume size is reached. Starting from the encrypted boot sector offset, all the

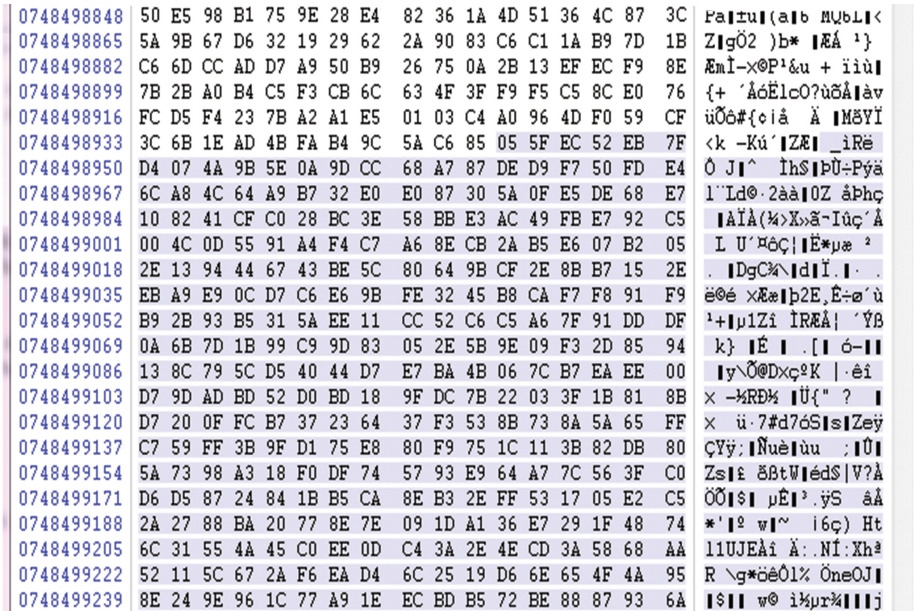
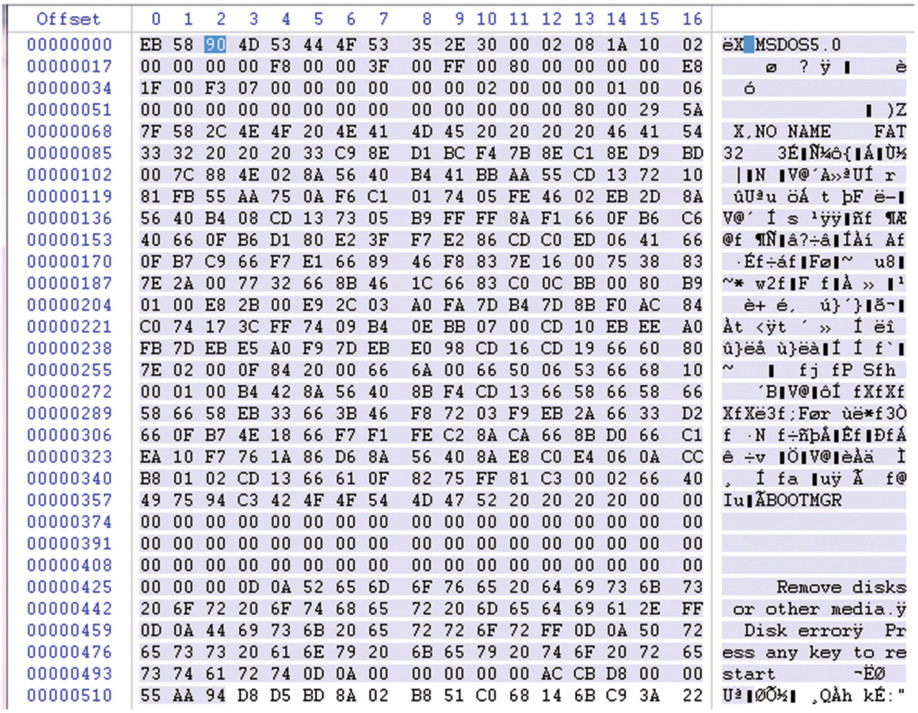


Fig. 7. Boot sector before decryption

sectors are decrypted and written in the same order starting from the boot sector until the encrypted size limit is reached. If the total number of sector is less than the encrypted volume size then there will be no problem. But if the total number of sectors is greater than the encrypted volume size, then the next corresponding sector should be the corresponding sector from the start of the volume header and so on.

For example, for the above evidence, if the volume header starts at sector 63, the original sector starts at sector 1404, the total number of sectors as 2506 and the encrypted volume size is 1009 sectors. Then while decrypting the sector 1404 (the original boot sector becomes the 63 sector), sector 1405 becomes sector 64 and so on up to sector 2413 since the encrypted volume size has reached. Then to get the sector 2414, we start at sector 63 and add up to the number of sectors covered (i.e. 1009). So the sector 1072 becomes the 2414 sector (63 (volume header sector) + 1009 (encrypted volume size)). In this manner, the sectors are decrypted and arranged to get the unbitlocked drive.



Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	08	1A	10	02	ëX MSDOS5.0
00000017	00	00	00	00	F8	00	00	3F	00	FF	00	80	00	00	00	00	E8	ø ? ÿ è
00000034	1F	00	F3	07	00	00	00	00	00	00	02	00	00	00	01	00	06	ó
00000051	00	00	00	00	00	00	00	00	00	00	00	00	00	80	00	29	5A	)Z
00000068	7F	58	2C	4E	4F	20	4E	41	4D	45	20	20	20	20	46	41	54	X,NO NAME FAT
00000085	33	32	20	20	20	33	C9	8E	D1	BC	F4	7B	8E	C1	8E	D9	BD	32 3É Ñ¼ô{ Á Ù½
00000102	00	7C	88	4E	02	8A	56	40	B4	41	BB	AA	55	CD	13	72	10	| N V@´A»ªUÍ r
00000119	81	FB	55	AA	75	0A	F6	C1	01	74	05	FE	46	02	EB	2D	8A	ûUªu öÁ t þF ë-
00000136	56	40	B4	08	CD	13	73	05	B9	FF	FF	8A	F1	66	0F	B6	C6	V@´ Í s ¹ÿÿ ñf ¶Æ
00000153	40	66	0F	B6	D1	80	E2	3F	F7	E2	86	CD	C0	ED	06	41	66	@f ¶Ñ â?÷â ÍÀí Af
00000170	0F	B7	C9	66	F7	E1	66	89	46	F8	83	7E	16	00	75	38	83	·Éf÷áf Fø ~ u8
00000187	7E	2A	00	77	32	66	8B	46	1C	66	83	C0	0C	BB	00	80	B9	~* w2f F f À » ¹
00000204	01	00	E8	2B	00	E9	2C	03	A0	FA	7D	B4	7D	8B	F0	AC	84	è+ é, ú}´}ð¬
00000221	C0	74	17	3C	FF	74	09	B4	0E	BB	07	00	CD	10	EB	EE	A0	Àt <ÿt ´ » Í ëî
00000238	FB	7D	EB	E5	A0	F9	7D	EB	E0	98	CD	16	CD	19	66	60	80	û}ëå ù}ëà Í Í f`
00000255	7E	02	00	0F	84	20	00	66	6A	00	66	50	06	53	66	68	10	~ fj fP Sfh
00000272	00	01	00	B4	42	8A	56	40	8B	F4	CD	13	66	58	66	58	66	´B V@ ôÍ fXfXf
00000289	58	66	58	EB	33	66	3B	46	F8	72	03	F9	EB	2A	66	33	D2	XfXë3f;Før ùë*f3Ò
00000306	66	0F	B7	4E	18	66	F7	F1	FE	C2	8A	CA	66	8B	D0	66	C1	f ·N f÷ñþÂ Êf Ðf Á
00000323	EA	10	F7	76	1A	86	D6	8A	56	40	8A	E8	C0	E4	06	0A	CC	ê ÷v ÖV@ èÀä Ì
00000340	B8	01	02	CD	13	66	61	0F	82	75	FF	81	C3	00	02	66	40	¸ Í fa uÿ Ã f@
00000357	49	75	94	C3	42	4F	4F	54	4D	47	52	20	20	20	20	00	00	Iu ÃBOOTMGR
00000374	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000391	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000408	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000425	00	00	00	0D	0A	52	65	6D	6F	76	65	20	64	69	73	6B	73	Remove disks
00000442	20	6F	72	20	6F	74	68	65	72	20	6D	65	64	69	61	2E	FF	or other media.ÿ
00000459	0D	0A	44	69	73	6B	20	65	72	72	6F	72	FF	0D	0A	50	72	Disk errorÿ Pr
00000476	65	73	73	20	61	6E	79	20	6B	65	79	20	74	6F	20	72	65	ess any key to re
00000493	73	74	61	72	74	0D	0A	00	00	00	00	00	AC	CB	D8	00	00	start ¬ËØ
00000510	55	AA	94	D8	D5	BD	8A	02	B8	51	C0	68	14	6B	C9	3A	22	Uª ØÕ½ ¸QÀh kÉ:"

Fig. 8. Boot sector after decryption

4 Conclusion

BitLocker To Go is a full-disk encryption protection technology for removable storage devices. Though it is based on BitLocker technology, BitLocker To Go significantly enhances the technical capabilities of BitLocker. A forensic examiner can use the recovery key or start key to access the FVEK and thus the protected data. These can be used to decrypt the series of keys protecting the FVEK like VMK. Some pieces of the metadata surrounding these keys could be useful to a forensic examiner, including the order in which keys were generated, the number of keys generated, and the types of those keys. Additionally, some features of the key management system allow access to all of the access devices protecting a volume provided the user has a valid access device.

References

1. Kumar, N., Kumar, V.: Bitlocker and Windows Vista, May 2008. http://www.nvlabs.in/node/9
2. Microsoft Corporation. Bitlocker drive encryption technical overview. Technical report, Microsoft Corporation, May 2008. http://technet2microsoft.com/WindowsVista/en/library/ce4d5a2e-59a5-4742-89cc-ef9f5908b4731033.mspx?mfr=true

3. Kornblum, J.D.: Implementing Bitlocker Drive Encryption For Forensic Analysis, ManTech International Corporation. jessekornblum.com/publications/di09.pdf
4. Metz, J.: Bitlocker Drive Encryption (BDE) format specification: Analysis of the BitLocker Drive Encryption (BDE) volume
5. Kornblum, J.D.: Bitlocker To Go, ManTech International Corporation. http://jessekornblum.com/presentations/dodcc10-1.pdf