Bitlocker and Windows Vista

This is an **INCOMPLETE draft version.**
Visit **www.nvlabs.in** for updates.

Nitin Kumar
Security Researcher
nitin.kumar@nvlabs.in

Vipin Kumar
Security Researcher
vipin.kumar@nvlabs.in

Contents

Abstract

This paper provides a complete technical overview of Microsoft® BitLocker™ Drive Encryption, a new data-protection feature in Microsoft Windows Vista™. The technical overview will cover all modes of operation of bitlocker and it functions.

This paper assumes that the reader understands Cryptography basics and Trusted Platform Model (TPM) technology.

This information applies for the Microsoft Windows Vista™ operating system.

The current version of this paper is maintained on the Web at: http://www.nvlabs.in/

References and resources discussed here are listed at the end of this paper.

Notes: Testing was performed on Windows Vista RC2 (Build 5744). Majority of the stuff remains valid for all versions of Windows Vista.

Requirements for Bitlocker

Bitlocker requires:-

1. TPM (Trusted Platform Module)
2. Capability to boot from USB drive

Now, we will go through the above in small details.

TPM :- Trusted Platform Module (TPM) is both the name of a published specification detailing a secure crypto-processor that can store secured information, as well as the general name of implementations of that specification, often called "TPM chip", "Fritz chip" or "TPM Security Device" (Dell). The TPM specification is the work of the Trusted Computing Group. The current version of the TPM specification is 1.2 Revision 103, published on July 9, 2007.

USB Drive:- The keys can also be written on a USB device instead of TPM. So, the system must support the reading and writing even without the support of drivers.

Bitlocker Modes of Operation

The following is the list of operational modes of Bitlocker

- Basic
  - TPM only :- all keys are stored within TPM
- Advanced
  - USB:- Key is stored on an external device
  - TPM + PIN:- TPM stores key together with a user specific PIN
  - TPM + USB:- TPM stores ½ key and USB stores another ½ half.
- TPM + USB + PIN (available in Vista SP1):- TPM stores ½ key, USB stores another ½ half, together with a user specific PIN.

Algorithms available in Bitlocker

Bitlocker uses one of the following algorithm to encrypt data

1. AES 128 bit
2. AES 128 bit + Diffuser
3. AES 256 bit
4. AES 256 bit + Diffuser

Diffuser is an additional stage designed to protect against ciphertext-manipulation attacks.

Keys in bitlocker

The following are the keys involved in bitlocker

1. VMK unlockers or VMK opening key (256 bit / 32 byte)
2. VMK key (256 bit / 32 byte)
3. FVEK key (128 bit/16 byte or 256 bit / 32 byte)
4. TWEAK key (128 bit/16 byte or 256 bit / 32 byte)
5. Sector key (256 bit / 32 byte)

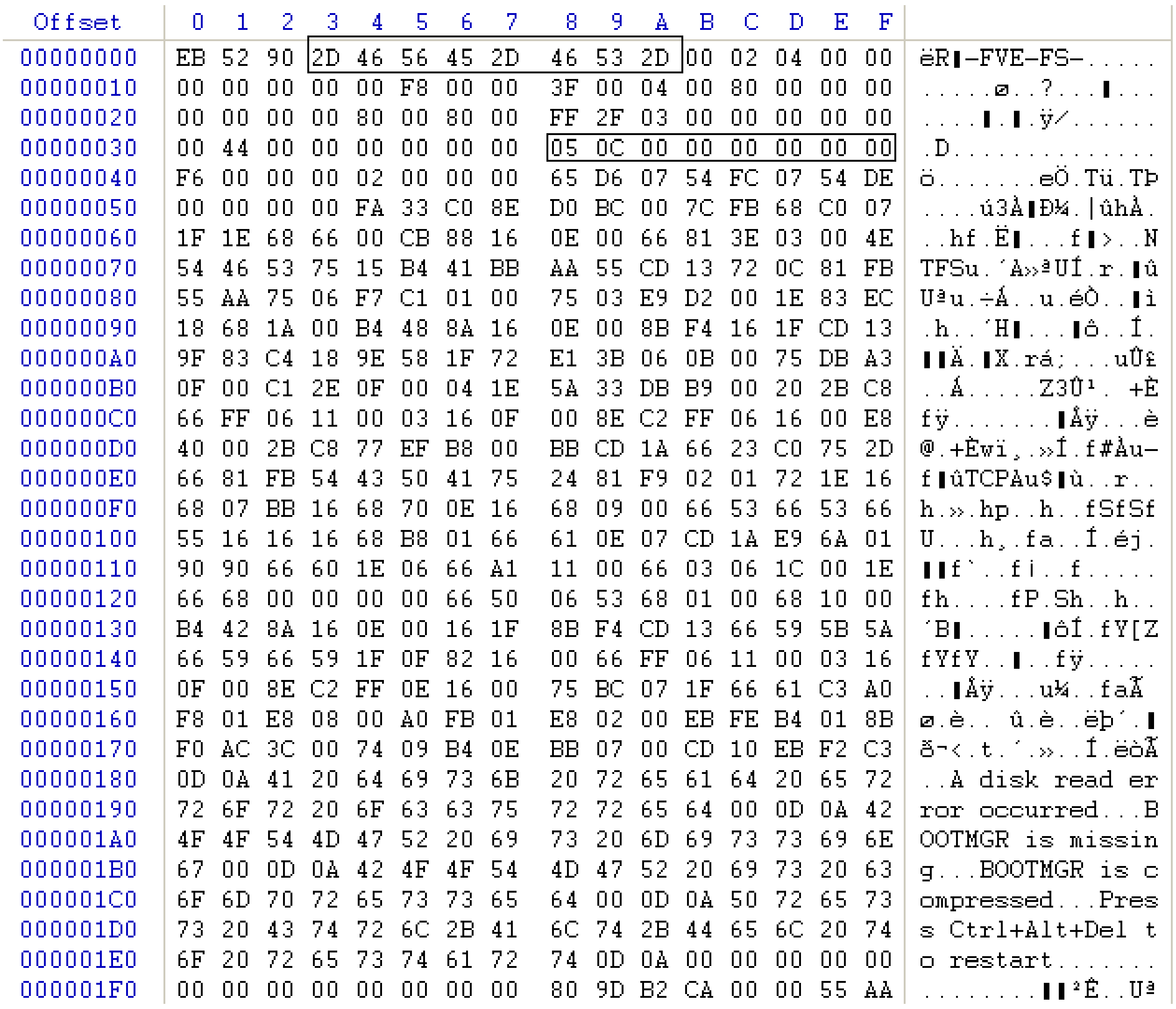
In case of different sizes of keys, the keys depend on the algorithm used to encrypt the volume.

The TWEAK key and Sector key are present and/or used only if diffuser is enabled.

Bit locker Volume Structure

Bitlocker Volume boot Sector or Sector 0 is 96 % similar to NTFS boot sector except the following changes

1. The Signature changes from "NTFS   " to "-FVE-FS-" at offset 3.[i]
2. The MFT_Mirror field at offset 0x38 is modified so as it now points to FVE_META_DATA.



```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  EB 52 90 2D 46 56 45 2D  46 53 2D 00 02 04 00 00  ëR▌-FVE-FS-.....
00000010  00 00 00 00 00 F8 00 00  3F 00 04 00 80 00 00 00  .....ø..?...▌...
00000020  00 00 00 00 80 00 80 00  FF 2F 03 00 00 00 00 00  ....▌.▌.ÿ/......
00000030  00 44 00 00 00 00 00 00  05 0C 00 00 00 00 00 00  .D..............
00000040  F6 00 00 00 02 00 00 00  65 D6 07 54 FC 07 54 DE  ö.......eÖ.Tü.TÞ
00000050  00 00 00 00 FA 33 C0 8E  D0 BC 00 7C FB 68 C0 07  ....ú3À▌Ð¼.|ûhÀ.
00000060  1F 1E 68 66 00 CB 88 16  0E 00 66 81 3E 03 00 4E  ..hf.Ë▌...f▌>..N
00000070  54 46 53 75 15 B4 41 BB  AA 55 CD 13 72 0C 81 FB  TFSu.´A»ªUÍ.r.▌û
00000080  55 AA 75 06 F7 C1 01 00  75 03 E9 D2 00 1E 83 EC  Uªu.÷Á..u.éÒ..▌ì
00000090  18 68 1A 00 B4 48 8A 16  0E 00 8B F4 16 1F CD 13  .h..´H▌...▌ô..Í.
000000A0  9F 83 C4 18 9E 58 1F 72  E1 3B 06 0B 00 75 DB A3  ▌▌Ä.▌X.rá;...uÛ£
000000B0  0F 00 C1 2E 0F 00 04 1E  5A 33 DB B9 00 20 2B C8  ..Á.....Z3Û¹. +È
000000C0  66 FF 06 11 00 03 16 0F  00 8E C2 FF 06 16 00 E8  fÿ.......▌Âÿ...è
000000D0  40 00 2B C8 77 EF B8 00  BB CD 1A 66 23 C0 75 2D  @.+Èwï¸.»Í.f#Àu-
000000E0  66 81 FB 54 43 50 41 75  24 81 F9 02 01 72 1E 16  f▌ûTCPAu$▌ù..r..
000000F0  68 07 BB 16 68 70 0E 16  68 09 00 66 53 66 53 66  h.».hp..h..fSfSf
00000100  55 16 16 16 68 B8 01 66  61 0E 07 CD 1A E9 6A 01  U...h¸.fa..Í.éj.
00000110  90 90 66 60 1E 06 66 A1  11 00 66 03 06 1C 00 1E  ▌▌f`..f¡..f.....
00000120  66 68 00 00 00 00 66 50  06 53 68 01 00 68 10 00  fh....fP.Sh..h..
00000130  B4 42 8A 16 0E 00 16 1F  8B F4 CD 13 66 59 5B 5A  ´B▌.....▌ôÍ.fY[Z
00000140  66 59 66 59 1F 0F 82 16  00 66 FF 06 11 00 03 16  fYfY..▌..fÿ.....
00000150  0F 00 8E C2 FF 0E 16 00  75 BC 07 1F 66 61 C3 A0  ..▌Âÿ...u¼..faÃ 
00000160  F8 01 E8 08 00 A0 FB 01  E8 02 00 EB FE B4 01 8B  ø.è.. û.è..ëþ´.▌
00000170  F0 AC 3C 00 74 09 B4 0E  BB 07 00 CD 10 EB F2 C3  ð¬<.t.´.»..Í.ëòÃ
00000180  0D 0A 41 20 64 69 73 6B  20 72 65 61 64 20 65 72  ..A disk read er
00000190  72 6F 72 20 6F 63 63 75  72 72 65 64 00 0D 0A 42  ror occurred...B
000001A0  4F 4F 54 4D 47 52 20 69  73 20 6D 69 73 73 69 6E  OOTMGR is missin
000001B0  67 00 0D 0A 42 4F 4F 54  4D 47 52 20 69 73 20 63  g...BOOTMGR is c
000001C0  6F 6D 70 72 65 73 73 65  64 00 0D 0A 50 72 65 73  ompressed...Pres
000001D0  73 20 43 74 72 6C 2B 41  6C 74 2B 44 65 6C 20 74  s Ctrl+Alt+Del t
000001E0  6F 20 72 65 73 74 61 72  74 0D 0A 00 00 00 00 00  o restart.......
000001F0  00 00 00 00 00 00 00 00  80 9D B2 CA 00 00 55 AA  ........▌▌²Ê..Uª
```

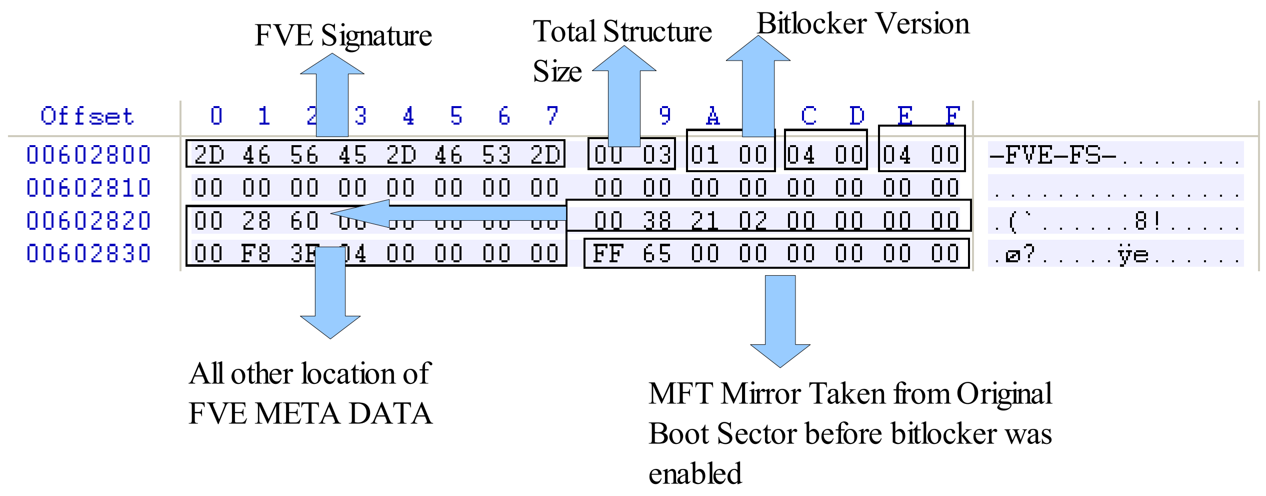
Illustration 1: Bitlocker Volume Boot Sector

FVE_META_DATA

FVE_META_DATA is the structure which contains all the information about keys,which keys are present and so on. It's a Control File if spoken in NTFS partition. Since nothing can be recovered from the partition, if the keys cannot be found, 3 copies are present and spread in the partition, so in case one gets damaged, another copy can be used.

The default size of Control File is 16,384 bytes.

Illustration 2: FVE META DATA First 64 bytes



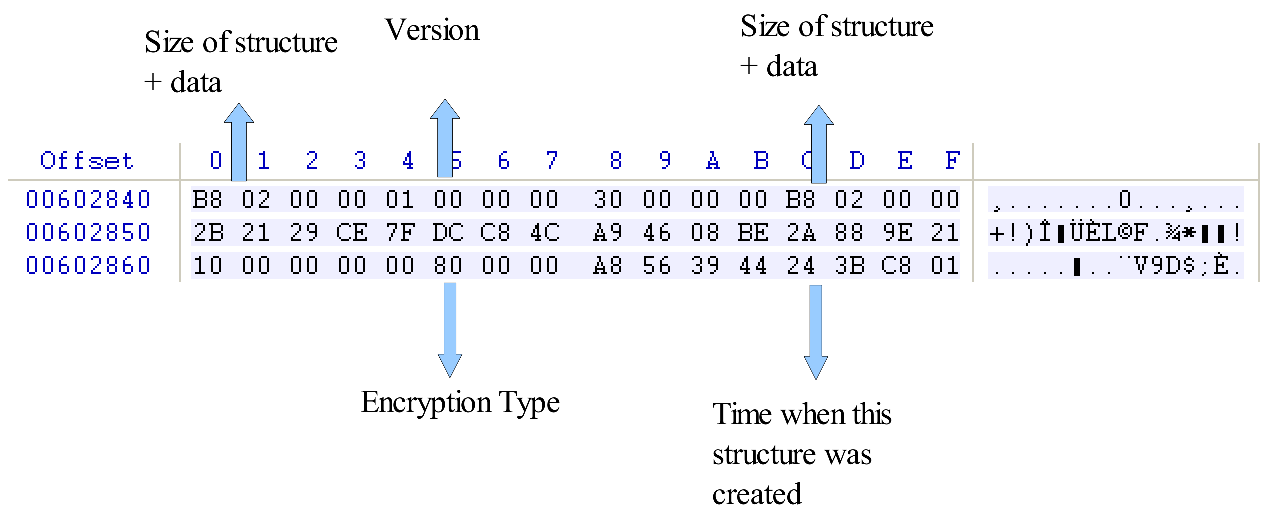
FVE Signature
Total Structure Size
Bitlocker Version
All other location of FVE META DATA
MFT Mirror Taken from Original Boot Sector before bitlocker was enabled

```
Offset    0  1  2  3  4  5  6  7     9  A     C  D  E  F
00602800 2D 46 56 45 2D 46 53 2D 00 03 01 00 04 00 04 00  -FVE-FS-........
00602810 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00602820 00 28 60 00 00 00 00 00 00 38 21 02 00 00 00 00  .(`.....8!......
00602830 00 F8 3F 04 00 00 00 00 FF 65 00 00 00 00 00 00  .ø?.....ÿe......
```

The above is represented in C code as

```
struct FVE_META_DATA {
     int8     Signature[8];
     int8     header[8];
     int8     reserved[16] ;
     int64    FVE_MetaData_LCN[3];
     int64    MFT_Mirror; // this value is filled from NTFS boot
sector before conversion
} MORE_FVE_DATA;
```

The next 48 bytes constitute the MAIN_HEADER. It contains the size of remaining data. This wraps the all encrypted VMK. However, it doesn't wrap the FVEK.



```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00602840  B8 02 00 00 01 00 00 00  30 00 00 00 B8 02 00 00  ¸.......0...¸...
00602850  2B 21 29 CE 7F DC C8 4C  A9 46 08 BE 2A 88 9E 21  +!)Î▌ÜÈL©F.¾*▌▌!
00602860  10 00 00 00 00 80 00 00  A8 56 39 44 24 3B C8 01  .....▌..¨V9D$;È.
```

```
In c code,
struct _MAIN_HEADER {
        int32    Size;   // Size of structure + size of data
        int32    Version; // it's 1  for now
        int32    Isize; // it is size of initial structure and is always 0x30
        int32     Size1; //a copy of Size
        int8    Reserved1[16]; // it's probably used to store something hash or salt but unknown,
        int32    Reserved2; // it's the same as Version
        int32    Encryption_Type; /* it's  should be one of these
                                       0x8000      AES 128 + diffuser
                                       0x8001      AES 256 + diffuser
                                       0x8002      AES 128
                                       0x8003      AES 256
                                       */
        FILETIME    Time;      // it is the time when this structure was
started to fill up so we can analyze this to found time or when it
was last updated and tell when was Bitlocker enabled
} MAIN_HEADER;
```

After this follow, a number of headers representing different data..The header size is 8 bytes and has the following structure.

```
struct _HEADER  {
     int16    Size;  // size of 8 byte + data
     int16    unknown1;
     int16    Type; // represents data followed by the header
     int16    Version; // it's 1
} HEADER;
```

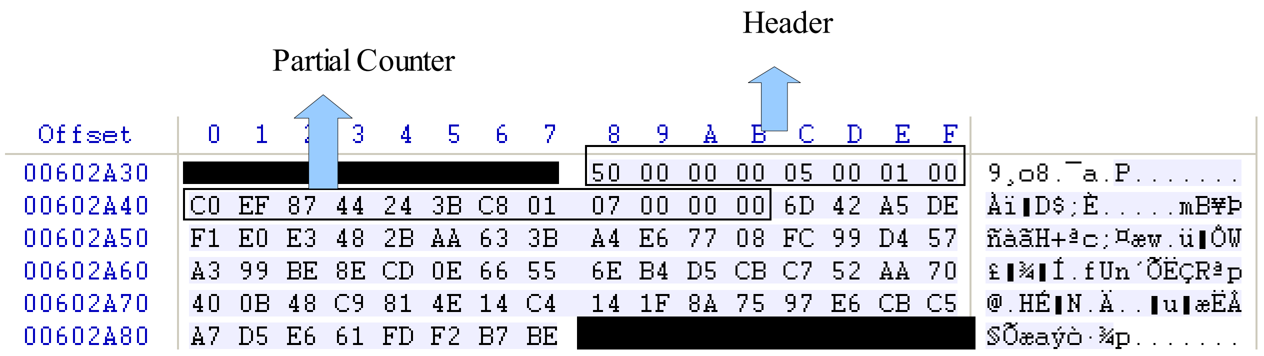
To jump from one header to another, read size , add it to current position and you are done.

The following header types are known

- 2 : the data structure is a unicode string
- 5: Encrypted key or data is stored
- 8 : Data structure is a key container

Bitlocker uses AES CCM mode to store keys. This provides both authentication and encryption. In this mode, a hash is created for the data to be stored. This hash is also stored and encrypted together with the data.Data is encrypted using AES-CBC mode.

Header Type 5: Encrypted Key or Data



```
 Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00602A30  [redacted]                50 00 00 00 05 00 01 00  9¸o8.¯a.P.......
00602A40  C0 EF 87 44 24 3B C8 01   07 00 00 00 6D 42 A5 DE  Àï▌D$;È.....mB¥Þ
00602A50  F1 E0 E3 48 2B AA 63 3B   A4 E6 77 08 FC 99 D4 57  ñàãH+ªc;¤æw.ü▌ÔW
00602A60  A3 99 BE 8E CD 0E 66 55   6E B4 D5 CB C7 52 AA 70  £▌¾▌Í.fUn´ÕËÇRªp
00602A70  40 0B 48 C9 81 4E 14 C4   14 1F 8A 75 97 E6 CB C5  @.HÉ▌N.Ä..▌u▌æËÅ
00602A80  A7 D5 E6 61 FD F2 B7 BE   [redacted]               §Õæaýò·¾p.......
```

Illustration 3: Sample Encrypted Key

AES requires counter to be 16 bytes. However, since only 12 bytes are present. This should be be completed

Completing the Counter

1 byte (15 - length of counter – 1) (so it evaluates to 2 for current version of bitlocker)

12 byte partial counter

2 byte 0

1 byte 0 (this is incremented after each block of encryption).

So, now we have a full 16 byte initialization vector.

The actual data is encrypted using CCM mode[ii] (Counter with CBC-Mode). It is an authenticated encryption algorithm designed to provide both authentication and privacy.

So, here is how it is done in Bitlocker.

Encryption:

First, a TAG(also called CBC-MAC) is created by repeated processing of the data, to get a final 16 byte block.

This block is now prepended to the actual data and is encrypted with AES 256 in Counter Mode, using the counter obtained as above.So, after encryption, block size increases by 16 bytes.

Decryption:

The whole encrypted data block is decrypted using the AES 256 in counter mode.Now the TAG (first 16 bytes) is extracted and a TAG(CBC-MAC) is computed from from the remaining data. If both the TAGs , the decrypted one and the one calculated from the decrypted data match, Data has been successfully decrypted.

VMK- Unlockers or Openers

A number of unlockers are present, such as recovery password, USB key, TPM etc.

Some of which would be dealt in detail

Recovery Password: It is 48 digit key which is used to decrypt volume,   in case of other methods fail or due to any other reason such as modification of boot environment or tampering with boot files.

SAMPLE Message and key

1. Save this numerical recovery password in a secure location away from your computer:

   471207-278498-422125-177177-561902-537405-468006-693451

   To prevent data loss, save this password immediately. This password helps ensure that you can unlock the encrypted volume.

Obtaining Key from Password:

Since this key only represents 128 bit, a key expansion procedure takes place. Which converts it into a full 256 bit which can the decrypt the FVEK.

Pseudocode

```
1. Divide each block by 11 , if the remainder is not 0 in all cases the key is not valid
2. collect the quotients from the above, and concatenate them to obtain a 128 bit key.
3. Take a 88 byte buffer  and zero it. The structure of the buffer is as follows
   // use 1 byte packing
   typedef struct {
   unsigned char sha_current[32];
   unsigned char sha_password[32];
   unsigned char salt[16];
   int64 hash_count;
           }blob;
4. Take SHA256 of the key and place it in the above structure in sha_password
5. The salt is place in the salt field of the above structure
6. Now run a loop  0x100000  ( 1048576)  times
7. Find SHA256 of the entire structure and place it in sha_current field
```

8. increment hash_count field counter in the structure
9. repeat steps 6 through 9 , till the loop is over
10. Take the first 32 bytes of the structure as the 256 bit key which can be used to decrypt the VMK corresponding to this key

USB key: in this form a key is stored on a USB drive which can unlock it's own encrypted VMK.

The file has a name such as 3926293F-E661-4417-A26C-C52286C5F149.BEK

BEK stands for Bitlocker Encryption Key(Most probably).

This file has a similar structure to that of MAIN_HEADER and thus will not be discussed here. However, the file ends with a KEY HEADER which will be followed by a 32 byte encryption key which will unlock the VMK. Also, the encryption and decryption has been discuused here, it will not be discussed here.

TPM: TPM is used to store the encryption key. TPM uses an internal key. This is how it works.

For complete working of TPM, [iii]

TPM has few basic operations such as generate key, decrypt,encrypt and clear operations.Also, the keys are locked with PCRs.

If the PCRs are correct, TPM will select required key and decrypt the data.

So, when Vista Boots, PCRs are automaticaly initialized, after BOOTMGR gains control, it supplies the encrypted VMK to TPM, and checks the result, if VMK could not be decrypted, BOOTMGR will display an error message, since system was tampered and a recovery password should be used to unlock the system and so on.

Completely discribing the TPM working is beyonf the scope of this paper[iv]

TPM + PIN : - In this combination , key is stored by the TPM together with user specific PIN input by the user. PIN is nothing both a 4-20 digit number. However before using the PIN, it should be expanded to 32 bytes. This is done by taking the SHA256 of the PIN. It is supplied to the TPM together with the encrypted VMK to get the decrypted VMK.
!! probably the key is xored with SHA256 has of PIN

TPM + USB: - This combination is similar to TPM + PIN except the data is stored on a USB key and expansion of key is .not required.

FVEK (Full Volume Encryption KEY)

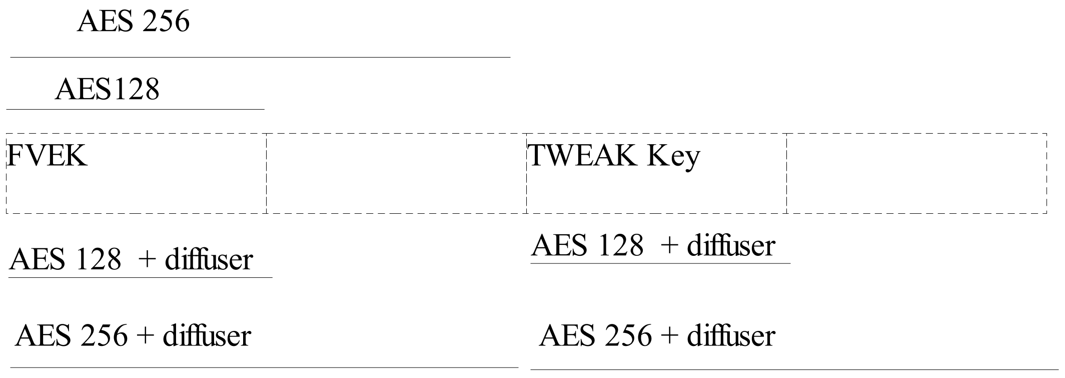
FVEK is unlocked by the VMK. On disk the the FVEK is stored in encrypted form using using AES CCM mode of encryption.

The size of FVEK depends upon the the algorithm chosen by the user.A default of AES 128 + diffuser is used

AES 128	128 bit key
AES 256	256 bit key
AES 128 + diffuser	512 bit key
AES 256 + diffuser	512 bit key

In case of diffuser, extra key called TWEAK key is also stored within the FVEK. If TWEAK key is present , it starts at boundary of 256 bits, so some bits of the FVEK can go unused in case of 128 bit algorithms

Division of FVEK 512 bits into parts



Data Encryption

Data is encrypted using AES. Most of it has been implemented in assembly, taking care of speed.

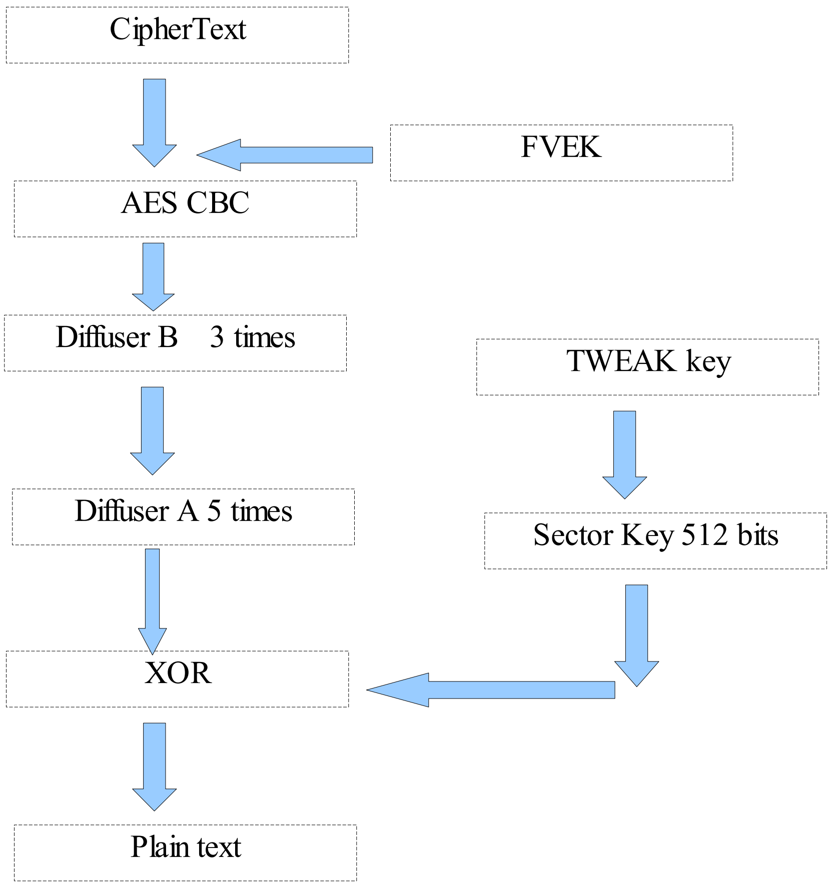
AES 128 and AES 256: In both these modes, data is encrypted using AES-CBC with zero initialization vector.

AES 128 + diffuser

AES 256 + diffuser:

except for the key size, both operate similar.

We will start with decryption because the encryption was analysed in decryption mode and encryption was obtained by reversing the stages of decryption.



The PseudoCode steps are

1. Decrypt Ciphertext with FVEK key 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 TWEAK key
5. XOR data obtained in step 3 with Sector Key
6. Plaintext

Now every step in the pseudocode will be detailed.

Step 1: Decrypt Ciphertext with FVEK key in AES-CBC mode

First we have to calculate Initialization Vector.To obtain IV, take the sector number in 64 bits in little endian order, also zero other 8 bytes, encrypt it with FVEK key.

Now do AES-CBC mode decryption with the IV obtained below

Step 2: Run Diffuser B in decryption direction 3 times

Diffuser B is already been documented by Microsoft. Read more at BitLockerCipher200608.pdf !!!!!!!!!!!!!!!!!!!!!!!!!!

Run diffuser B 3 times in decryption direction.

Step 3: Run Diffuser A in decryption direction 5 times

Diffuser A is also been documented by Microsoft.

Run diffuser A 5 times in decryption direction.

Step 4: Calculate Sector Key from TWEAK key

Take a buffer of 16 bytes, zero it.Now copy the Sector Number in little endian format and encrypt it with TWEAK key to obtain first 16 bytes of Sector key.

Take a buffer of 16 bytes, zero it.Now copy the Sector Number in little endian format and make the 16th byte as 128 or 0x80,now encrypt it with TWEAK key to obtain remaining 16 bytes of Sector Key. Concatenate both part to obtain full 32 byte or 512 bit Sector Key.

Step 5: XOR data obtained in step 3 with Sector Key

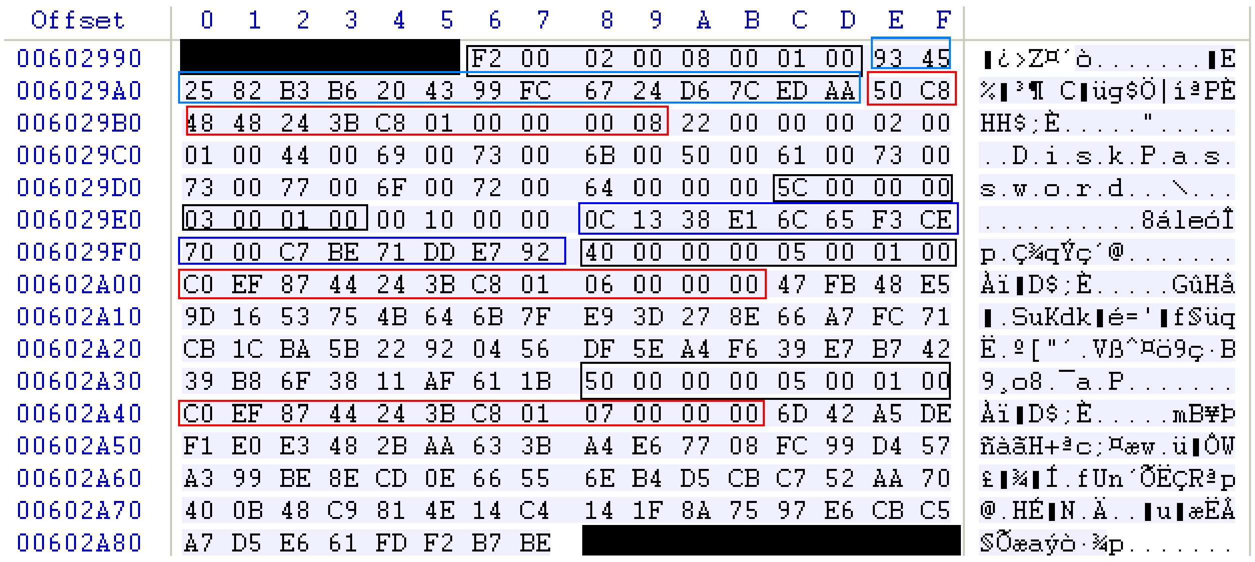
XOR the data with Sector key. Since data is minimum of 512 bytes , repeat sector key   as many times as necessary.

Step 6: Plaintext

Whatever, you have now is the plaintext

48 digit Recovery Password

VMK structure



```
Offset      0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00602990   [redacted]        F2 00   02 00 08 00 01 00 93 45   ▌¿›Z¤´ò.......▌E
006029A0   25 82 B3 B6 20 43 99 FC   67 24 D6 7C ED AA 50 C8   %▌³¶ C▌üg$Ö|íªPÈ
006029B0   48 48 24 3B C8 01 00 00   00 08 22 00 00 00 02 00   HH$;È....."..... 
006029C0   01 00 44 00 69 00 73 00   6B 00 50 00 61 00 73 00   ..D.i.s.k.P.a.s.
006029D0   73 00 77 00 6F 00 72 00   64 00 00 00 5C 00 00 00   s.w.o.r.d...\...
006029E0   03 00 01 00 00 10 00 00   0C 13 38 E1 6C 65 F3 CE   ..........8áleóÎ
006029F0   70 00 C7 BE 71 DD E7 92   40 00 00 00 05 00 01 00   p.Ç¾qÝç´@.......
00602A00   C0 EF 87 44 24 3B C8 01   06 00 00 00 47 FB 48 E5   Àï▌D$;È.....GûHå
00602A10   9D 16 53 75 4B 64 6B 7F   E9 3D 27 8E 66 A7 FC 71   ▌.SuKdk▌é='▌f§üq
00602A20   CB 1C BA 5B 22 92 04 56   DF 5E A4 F6 39 E7 B7 42   Ë.º["´.Vß^¤ö9ç·B
00602A30   39 B8 6F 38 11 AF 61 1B   50 00 00 00 05 00 01 00   9¸o8.¯a.P.......
00602A40   C0 EF 87 44 24 3B C8 01   07 00 00 00 6D 42 A5 DE   Àï▌D$;È.....mB¥Þ
00602A50   F1 E0 E3 48 2B AA 63 3B   A4 E6 77 08 FC 99 D4 57   ñàãH+ªc;¤æw.ü▌ÔW
00602A60   A3 99 BE 8E CD 0E 66 55   6E B4 D5 CB C7 52 AA 70   £▌¾▌Í.fUn´ÕËÇRªp
00602A70   40 0B 48 C9 81 4E 14 C4   14 1F 8A 75 97 E6 CB C5   @.HÉ▌N.Ä..▌u▌æËÅ
00602A80   A7 D5 E6 61 FD F2 B7 BE   [redacted]                §Õæaýò·¾p.......
```

i For more information about NTFS partition structure, use http://www.ntfs.com/ntfs-partition-boot-sector.htm
ii http://en.wikipedia.org/wiki/CCM_mode
iii TPM specification wiki
iv TPM specification