The Analysis of Windows Vista Disk Encryption Algorithm

Mohamed Abo El-Fotouh and Klaus Diepold

Institute for Data Processing (LDV)
Technische Universität München (TUM)
80333 Munich Germany
mohamed@tum.de
kldi@tum.de

Abstract. Windows Vista Enterprise and Ultimate editions use Bitlocker Drive Encryption as its disk encryption algorithm, and at its heart is the AES-CBC + Elephant diffuser encryption algorithm (ELEPHANT). In this paper we present our analysis of ELEPHANT using statistical tests. Our analysis has explored some weaknesses in its diffusers, thus we propose new diffusers to replace them. The new diffusers overcome the weaknesses of the original ones, and offer better and faster diffusion properties. We used the new diffusers to build variants of ELEPHANT, that possess better diffusion properties.

Keywords: Disk encryption, Windows Vista disk encryption algorithm.

1 Introduction

Data security on lost or stolen PCs is a growing concern among security experts and corporate executives. The data stored on the PC asset is often significantly more valuable to a corporation than the asset itself, and the loss, theft or unwanted disclosure of that data can be very damaging. Thus, this data should be encrypted to minimize that loss. Disk encryption applications are used to encrypt all the data on the hard disk, where all the hard disk is encrypted with a single/multiple key(s) and encryption/decryption are done on the fly, without user interference.

Disk encryption usually encrypts/decrypts a whole sector at a time. There exist dedicated block ciphers, that encrypts a whole sector at once. Bear, Lion, Beast and Mercy [1, 1, 2, 3] are examples of these ciphers. Bear, Lion and Beast are considered to be slow, as they pass the data multiple times and Mercy was also broken in [4]. The other method is to let a block cipher like the AES [5] (with 16 bytes block size) to process the data within a mode of operation. The most used mode of operation is CBC [6], but it is subjected to manipulation attacks. There exist other modes of operations dedicated to solve this problem XTS, XCB, CMC and EME [7, 8, 9, 10] are to name a few.

The Enterprize and Ultimate editions of Windows Vista contain a new feature called Bitlocker Drive Encryption which encrypts all the data on the system

volume [11]. Bitlocker uses existing technologies like the AES in the CBC mode and TPM [12], together with two new diffusers.

In this paper, we study the current implementation of AES-CBC + Elephant diffuser (ELEPHANT) and propose new diffusers to replace its diffusers. The proposed diffusers possess better and faster diffusion properties than the current ones. We used the proposed diffusers to construct two variants of ELEPHANT. Our study shows that the proposed diffusers and variants of ELEPHANT, possess better diffusion properties.

In section 2, we describe ELEPHANT with its current diffusers (CURDIFF). In section 3, we propose new diffusers (NEWDIFF) and two variants of ELEPHANT which we name NEWELF and NEWELFRED. In section 4, we tried to answer the following questions: Does the cipher/diffuser behave randomly as expected with different patterns of plaintexts and tweaks? How sensitive is the cipher/diffuser to a change in the plaintext/tweak? We examined different data-sets against randomness to answer these questions. In section 5, we tried to answer the following questions: Can the cipher be reduced to CBC? Is the tested cipher correlated with CBC? Does the cipher/diffuser suffer from the bit-flipping attack? We designed statistical test to answer these questions. In section 6, we tried to answer the following questions: Does the cipher possess the avalanche effect in the encryption direction? Does the cipher possess poor man's authentication property [11]? We designed statistical test to answer these questions. In section 7, we tried to answer the following questions: Does each bit in ciphertext depend on all the bits in the plaintext? Does each bit in plaintext depend on all the bits in the ciphertext? We designed statistical test to answer these questions. We present a performance analysis of the ciphers/diffusers in section 8, and our discussion in section 9 and finally we conclude in section 10.

2 Current implementation

2.1 ELEPHANT

Figure 1 shows an overview of ELEPHANT [11]. There are four steps to encrypt a sector:

1. The plaintext is xored with a sector key K_s (1).
2. The result of the previous step run through diffuser A.
3. The result of the previous step run through diffuser B.
4. The result of the previous step is encrypted with AES in CBC mode using IV_s (2), as the initial vector.

$$K_s = E(K_{sec}, e(s)) \parallel E(K_{sec}, e'(s)) \tag{1}$$

$$IV_s = E(K_{AES}, e(s)) \tag{2}$$

Where E() is the AES encryption function, K_{sec} is a key used to generate K_s, K_{AES} is the key used to generate the sector IV_s and used in the AES-CBC process, e() is an encoding function that maps each sector number s into a unique

16-byte value. The first 8 bytes of the result are the byte offset of the sector on the volume. This integer is encoded in least-significant-byte first encoding. The last 8 bytes of the result are always zero and e'(s) is the same as e(s) except that the last byte of the result has the value 128.
Note that the plaintext and key are parameterized. In our study we used the following parameters:

1. Plaintext of size 4096-bits (the current standard sector size).
2. Tweak-Key of size 384-bits (the first 128-bits serves as the IV_s "Sector Initial Vector" for the AES-CBC and the other 256-bits serve as K_s "Drive Sector Key").
3. We examined the 256-bits key version of the AES (that provides maximum security), that means both K_{sec} and K_{AES} are of size 256-bits.

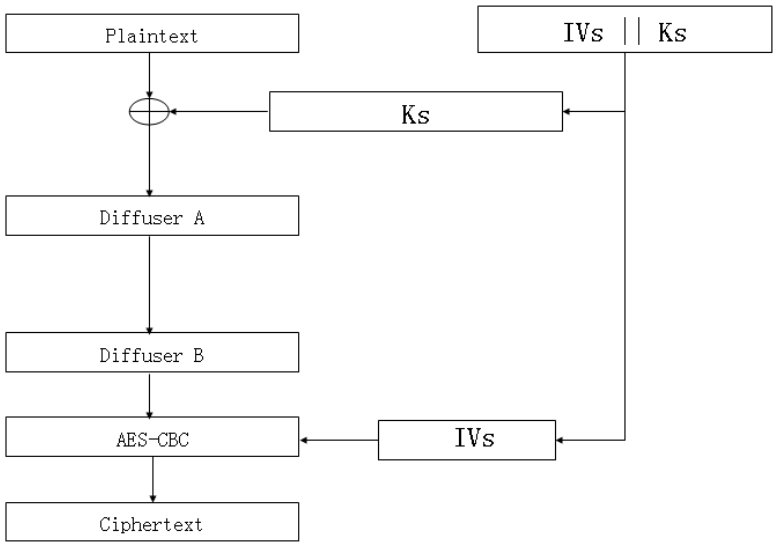

Fig. 1. Overview of AES-CBC with Elephant Diffuser.

2.2 The Diffusers

The current diffusers (CURRDIFF) are very similar. The following notations are used to define the diffusers:

1. d_i is the i^{th} 32-bits word in the sector, if i falls outside the range then $d_i = d_{i \bmod n}$, where n is the number of the 32-bits in the sector.

2. AC and BC are the number of cycles of diffuser A and B, they are defined to be 5 and 3 respectively.
3. RA = [9, 0, 13, 0] and RB = [0, 10, 0, 25] hold the rotation constants of diffuser A and B respectively.
4. $\oplus$ is the bitwise xor operation.
5. << is the integer 32-bit left rotation operation, where the rotation value is written on its right size.
6. - is integer subtraction modulo 2^{32}.

Table 1 presents the description of the CURRDIFF (diffuser A and diffuser B).

Table 1. Current diffusers.

Diffuser A:	**Diffuser B:**
for j=1 to AC	for j=1 to BC
for i=n-1,...,2,1,0	for i= n-1,...,2,1,0
t=(d_{i-5} << $RA_{i \bmod 4}$)	t=(d_{i+5} << $RB_{i \bmod 4}$)
t=t $\oplus$ d_{i-2}	t=t $\oplus$ d_{i+2}
d_i= d_i - t	d_i= d_i - t

3 Proposed Modification

The novelty of this study is to modify ELEPHANT to possess better and faster diffusion properties, we have replaced diffuser A and B with diffuser A' and B'. We named the current implementation of the diffuser layer (diffuser A followed by diffuser B, where AC=5 and BC=3) thorough out our study CURDIFF and our proposed diffuser layer NEWDIFF (diffuser A' followed by diffuser B', where AC=5 and BC=3). We propose a variant of ELEPHANT, we call it NEWELF. It is the same as ELEPHANT after replacing CURRDIFF with NEWDIFF. We also propose NEWELFRED, which is a variant of NEWELF, where it uses reduced number of rounds (AC=1 and BC=2).

3.1 Motivation

From studying the current diffusers, three undesired properties have been found:

1. If their input is of all zeros or of all ones, their output will be identical to their input. This is true for both the encryption and decryption directions. This is due to the fact, that the result of the xor operations (in diffuser A and diffuser B) will always be zero and the diffusers are bypassed (i.e. that sector will be encrypted with CBC only). This is due to the absence of any confusion operations.
2. The current diffusers are completely linear functions, that do not offer any form of non-linearity. Due to the absence of confusion operations.
3. The current diffusers updates only a single word (in the inner loop), thus the diffusion is slow.

3.2 Proposed diffusers

The main objectives of the proposed diffusers are to overcome the limitations of the current diffusers. Table 2 presents the description of the NEWDIFF (diffuser A' and diffuser B'), where SBOX[X] returns 8-bits from the AES SBOX, using the least significant 8-bits of X as the index.

Table 2. Proposed diffusers.

Diffuser A':	**Diffuser B':**
for j=1 to AC	for j=1 to BC
for i=n-1,...,2,1,0	for i= n-1,...,2,1,0
$d_{i-5} = d_{i-5} \oplus SBOX[d_i]$	$d_{i+5} = d_{i+5} \oplus SBOX[d_i]$
$d_{i-5} = d_{i-5} << RA_{i \bmod 4}$	$d_{i+5} = d_{i+5} << RA_{i \bmod 4}$
$d_{i-5} = d_{i-5} \oplus d_{i-2}$	$d_{i+5} = d_{i+5} \oplus d_{i+2}$
$d_i = d_i - d_{i-5}$	$d_i = d_i - d_{i+5}$

3.3 Discussion

The proposed diffusers possess the following properties:

1. They can not be easily bypassed, like the current diffusers. Thanks to the SBOX which offers confusion.
2. The confusion operation is well studied (AES SBOX) and they offer good confusion properties. Note that, all the non-linearity of the AES is offered by its SBOX [13].
3. Two 32-bits words are updated in the inner loop of the diffusers, thus providing faster diffusion properties (see Sect. 8), for example for diffuser A':
   (a) d_{i-5} is first xored with the result of SBOX of d_i, that means the last 8-bits of d_{i-5} depends on each bit in the last 8-bits of d_i.
   (b) Then the rotation performs diffusion within d_{i-5}, which reflect the effect of the previous step.
   (c) Then d_{i-5} is xored with d_{i-2}, so each corresponding bit of d_{i-5} depends of that of d_{i-2}.
   (d) Finally d_{i-5} is subtracted from d_i, which means that each corresponding bit of d_i depends of that of d_{i-5} (reflecting the effects of all the previous steps).

In the next sections we are going to present different statistical tests and their corresponding results . We divide these tests into four different categories, each category tries to answer specific questions, to help us better understand the behavior of the tested ciphers/diffusers.

4 Randomness tests

One of the criteria used to evaluate block ciphers is their demonstrated suitability as random number generators. That is, the evaluation of their outputs utilizing statistical tests should not provide any means by which to computationally distinguish them from truly random sources [14]. In [15], the randomness of the final five candidates of the AES algorithms were tested. Another study [16], which we applied here, applies the NIST statistical tool [17] to the disk encryption modes of operation, where eleven data-sets are subjected to 188 statistical tests each. These tests try to explore the behavior of the ciphers/diffusers for different patterns of tweak and plaintext values, these data-sets are:

1. Random plaintext / random tweak.
2. Random plaintext / low density tweak.
3. Random plaintext / high density tweak.
4. Low density plaintext / random tweak.
5. Low density plaintext / low density tweak.
6. Low density plaintext / high density tweak.
7. High density plaintext / random tweak.
8. High density plaintext / low density tweak.
9. High density plaintext / high density tweak.
10. Plaintext avalanche.
11. Tweak avalanche.

For more details about these data-sets please refer to [16]. In table 3 we reported the number of failed tests (out of 188) for each cipher/diffuser for the eleven data sets, where a test fails when either the cipher/diffuser failed that test too often or the output is uniform. These tests try to answer the following questions: *Does the cipher/diffuser behave randomly as expected with different patterns of plaintexts and tweaks? How sensitive is the cipher/diffuser to a change in the plaintext/tweak?* ELEPHANT, NEWELF and NEWELFRED possess a good random profile, while CBC possesses an acceptable random profile (it has problems with plaintext avalanche test, which is expected as it pass the data only once). The proposed NEWDIFF possesses a good random profile, however CURRDIFF possesses a weak one (CURRDIFF fails completely when the plaintext is a repeated pattern and it is not so sensitive to a tweak change). Note that as the tweak (K_s) can not be all zero or all ones, refer to(1), to produce a tweak that is low/high density. We ran the tests two time, one with the first half low/high density and the rest random, the second time with the first half random and the second half with low/high density.

5 Correlation tests

5.1 CBC-Correlation function

As ELEPHANT, NEWELF and NEWELFRED are based on CBC, we measured their correlation with CBC, using the nine combinations of all zero, all one, and

Table 3. Number of failed statistical tests for the eleven data-sets.

Data set #	1	2	3	4	5	6	7	8	9	10	11
CBC	10	13	11	15	22	19	16	16	18	**166**	30
ELEPHANT	15	10	9	20	7	17	12	12	12	12	15
	12	7	17	22	17	15	14	12	14	12	33
CURRDIFF	18	18	18	**185**	**187**	**187**	**173**	**187**	**187**	9	**186**
	18	18	18	**185**	**187**	**187**	**174**	**187**	**187**	9	**186**
NEWDIFF	8	6	13	16	12	17	10	14	16	6	13
	9	7	7	20	12	15	12	11	15	8	34
NEWELF	18	10	9	8	16	12	9	12	19	10	8
	9	10	12	15	14	15	9	15	15	12	33
NEWELFRED	8	9	10	8	12	15	18	13	17	8	30
	12	7	13	11	11	8	15	16	12	7	13

all random bits between the plaintext and the tweak. This function is called **CBC-Correlation**. This function tries to answer the following question: *Can the cipher be reduced to CBC?* NEWELF and NEWELFRED succeeded to pass **CBC-Correlation** function, while ELEPHANT failed to pass the test for two inputs:

1. When the tweak is all zeros and the plaintext is all zeros.
2. When the tweak is all zeros and the plaintext is all ones.

In both cases the diffuser layer has **no effect** on the plaintext. Although with the current design of the tweak, it is impossible to get a tweak with all zeros refer to (1), it is still possible to bypass the CURRDIFF in those two cases:

1. when the encrypted sector contains the repetitions of K_s (i.e. the xor operation will result in all zero plaintext and the diffuser layer will be bypassed).
2. when the encrypted sector contains the repetitions of the negation of K_s (i.e. the xor operation will result in all ones plaintext and the diffuser layer will be bypassed).

These failures are due to the absence of non-linear operations in CURRDIFF and that the result of the xor operations in CURRDIFF will result always with zero (the identity element of subtraction), when the input is all zeros or all ones. In these two cases ELEPHANT is reduced to CBC.

5.2 Bit-flipping Attack

We applied the Bit-Flipping-Attack function three times each with a different parameter x, ($0 \leq x \leq 2$) that determines the pattern of the used plaintext and tweak, the function is listed in table 5 and its nomenclatures are in table 4. Bit-Flipping-Attack function tests if changing any bit in the ciphertext will be associated with changing a specific bit(s) in the plaintext. This function tries

to answer the following question: *does the cipher/diffuser suffer from the bit-flipping attack ?* If the maximum returned by the "Summary" function is equal to the sample size, that means there is at least one bit that changes whenever a specific bit in the ciphertext is changed, and that means bit-flipping attack [18] is applicable on the tested cipher/diffuser. The results of the Bit-Flipping-Attack

Table 4. Nomenclatures for some test functions.

GenRndKey(X)	Generates a random key X with length 256-bits.
No_of_Bits	The number of bits in the sector.
Samplesize	Number of random samples used for each bit location, we used 1539 samples.
Init(R)	Initialize the array R with zeros.
GenerateSector(P)	Generates a sector P of size 4096-bits, for the first 513 calls it returns low density plaintext, for the next 513 calls it returns high density plaintext and for the last 513 calls it returns random plaintext.
GenTweak(K2,x)	Generates a 384-bits tweak. If x equals zero then all the tweak is filled with zero bits, if x equals ones then all the tweak is filled with one bits, otherwise the tweak is filled with random bits.
Encrypt(P,C,A,B)	Encrypts the plaintext P to the ciphertext C, using A as the encryption key and B as the tweak.
ChangeBit(C,i,C2)	Flips the bit number i in the text C and put the result in C2.
Decrypt(C,P,A,B)	D ecrypts the ciphertext C to the plaintext P, using A as the encryption key and B as the tweak.
Xor(R,P,T)	Xors P with T and puts the results in R.
Add(R,Y)	Adds the values in the array Y to that in the array R.
Analyze(R,Matrix[i])	Calculates the minimum, maximum, average and standard deviation of the R array. The above four values as stored in Matrix[i]. Note: any one added to the R array represents that this bit has changed as a result of changing the bit number i in the ciphertext/plaintext.
Summarize(Matrix)	Calculates the normalized minimum of minimums **(Min)**, maximum of the maximums **(Max)**, average of the averages **(AVG)** and average of the standard deviations **(SD)** in Matrix.

in table 6, show that ELEPHANT, NEWELF, NEWELFRED, CURRDIFF, NEWDIFF pass these tests, while CBC fails these tests as it is subjected to the bit-flipping attack.

Table 5. Bit-Flipping-Attack function.

```
Function Bit-Flipping-Attack(x)
double Martrix[No_of_Bits][Samplesize];
GenRndKey(K1);
GenTweak(K2,x);
For (i=0;i<No_of_Bits;i++)
{      Init(R);
      For(j=0;j<Samplesize;j++)
            {GenSector(P);
             Encrypt(P,C,K1,K2);
             ChangeBit(C,I,C2);
             Decrypt(C2,T,K1,K2);
             Xor(Y,P,T);
             Add(R,Y) ;}
      Analyze(R,Matrix[i]);
}
Summarize(Matrix);
```

Table 6. Bit-Flipping-Attack results.

	ELEPHANT				**CBC**			
x	Min	Max	AVG	SD	Min	Max	AVG	SD
0	0.23	0.76	0.5	0.05	0	1	0.02	0.09
1	0.24	0.76	0.5	0.05	0	1	0.02	0.09
2	0.24	0.78	0.5	0.05	0	1	0.02	0.09
	CURRDIFF				**NEWDIFF**			
x	Min	Max	AVG	SD	Min	Max	AVG	SD
0	0.24	0.78	0.5	0.05	0.25	0.77	0.5	0.05
1	0.22	0.76	0.5	0.05	0.24	0.78	0.5	0.05
2	0.24	0.75	0.5	0.05	0.24	0.79	0.5	0.05
	NEWELF				**NEWELFRED**			
x	Min	Max	AVG	SD	Min	Max	AVG	SD
0	0.24	0.78	0.5	0.05	0.24	0.79	0.5	0.05
1	0.22	0.77	0.5	0.05	0.25	0.75	0.5	0.05
2	0.25	0.76	0.5	0.05	0.25	0.77	0.5	0.05

6 Avalanche tests

This category consists of six tests, where two functions are applied three times with a different parameter x, ($0 \leq x \leq 2$), that determines the pattern of the used plaintext and tweak, these two functions are :

- **Avalanche-Encryption(x)**: measures avalanche effect [19] in the encryption direction (the effect of changing one bit of plaintext on the ciphertext), a good cipher will have roughly half the bits of the ciphertext changed due to a single bit change in plaintext. It tries to answer the following question: *does the cipher possess the avalanche effect in the encryption direction?*
- **Avalanche-Decryption(x)**: measures avalanche effect in the decryption direction (the effect of changing one bit of ciphertext on the plaintext), this is to possess poor man's authentication, that is changing one bit in the ciphertext will lead that the plaintext will be scrambled. It tries to answer the following question: *does the cipher possess poor man's authentication property?*

The results in tables 7 and 8 show that ELEPHANT, NEWELF, NEWELFRED, CURRDIFF, NEWDIFF all have good avalanche effect in both encryption and decryption directions, on the other hand CBC failed to pass these tests.

Table 7. Avalanche-Encryption results.

	ELEPHANT				**CBC**			
x	Min	Max	AVG	SD	Min	Max	AVG	SD
0	0.46	0.54	0.5	0.01	0.01	0.99	0.5	0.28
1	0.46	0.54	0.5	0.01	0.01	0.99	0.5	0.28
2	0.47	0.54	0.5	0.01	0.01	0.99	0.5	0.28
	CURRDIFF				**NEWDIFF**			
x	Min	Max	AVG	SD	Min	Max	AVG	SD
0	0.46	0.53	0.5	0.01	0.46	0.53	0.5	0.01
1	0.47	0.54	0.5	0.01	0.46	0.54	0.5	0.01
2	0.46	0.54	0.5	0.01	0.46	0.53	0.5	0.01
	NEWELF				**NEWELFRED**			
x	Min	Max	AVG	SD	Min	Max	AVG	SD
0	0.46	0.54	0.5	0.01	0.46	0.54	0.5	0.01
1	0.46	0.54	0.5	0.01	0.46	0.54	0.5	0.01
2	0.46	0.53	0.5	0.01	0.46	0.54	0.5	0.01

7 Bit dependency tests

This category consists of two tests :

Table 8. Avalanche-Decryption results.

	ELEPHANT				**CBC**			
x	Min	Max	AVG	SD	Min	Max	AVG	SD
0	0.46	0.54	0.5	0.01	0.01	0.99	0.49	0.48
1	0.47	0.54	0.5	0.01	0.01	0.99	0.5	0.48
2	0.46	0.54	0.5	0.01	0.01	0.99	0.5	0.48
	CURRDIFF				**NEWDIFF**			
x	Min	Max	AVG	SD	Min	Max	AVG	SD
0	0.46	0.53	0.5	0.01	0.46	0.53	0.5	0.01
1	0.46	0.53	0.5	0.01	0.47	0.54	0.5	0.01
2	0.46	0.53	0.5	0.01	0.46	0.54	0.5	0.01
	NEWELF				**NEWELFRED**			
x	Min	Max	AVG	SD	Min	Max	AVG	SD
0	0.46	0.54	0.5	0.01	0.46	0.54	0.5	0.01
1	0.46	0.54	0.5	0.01	0.46	0.54	0.5	0.01
2	0.46	0.54	0.5	0.01	0.47	0.54	0.5	0.01

- **BD-Encryption()**: is passed, when each bit in the ciphertext depends on every bit in the plaintext. It tries to answer: *does each bit in ciphertext depend on all the bits in the plaintext?*
- **BD-Decryption()**: is passed, when each bin in the plaintext depends on every bit in the ciphertext. It tries to answer: *does each bit in plaintext depend on all the bits in the ciphertext?*

The Bit-dependency functions are measured as following:

1. A dependency matrix M is constructed of size B × B (where B is the number of bits in the plaintext/ciphertext, here B = 4096).
2. The diagonal is initialized by 1 and all other bits are set to zero, as initially each bit depends only on itself.
3. Depending on the applied operations the matrix M is updated, BD-Encryption applies the operation in the encryption direction and BD-Decryption applies them in the decryption direction.
4. If an output bit is dependent on an input bit(s), the column of the output bit is ORed with that (those) of the input bit(s). For example:
   (a) Xor operation: each output bit is dependent on the corresponding input bit.
   (b) Addition and subtraction modulo 2^{32} operations are approximated to an xor operation for simplicity and generality.
   (c) AES operation: each bit in the input 128-bits is dependent on the other 127 bits.
   (d) 32-bit rotation: the columns change there order depending on the rotation amount and direction.
   (e) SBOX look up: each bit of the output depends on every bit of the input.

5. All the operation of the tested function/cipher are applied and the matrix M is updated.
6. At the end the sum of all ones in the matrix is calculated and is divided by B^2.
7. If the returned value in the previous step is 1, this means that each bit of the output bits depends on all the bits of the input and the function succeeds, it fails otherwise.

The results of applying **BD-Encryption** and **BD-Decryption** functions are found in table 9, where we reported the minimum values of AC and BC each algorithm needs to pass these tests (under columns AC' and BC'), together with the current used values. The results show that all the ciphers but CBC succeeded these tests. CURDIFF needs at least three rounds of diffuser B and two rounds of diffuser A, on the other hand ELEPHANT which uses it needs only at least AC=2 and BC=1 to pass it, this is because the CBC layer does the rest of the diffusion. NEWDIFF needs at least AC=1 and BC=2, while NEWELF and NEWELFRED needs only BC=1 and the CBC layer does the rest of the diffusion.

Table 9. BD-Encryption and BD-Decryption results.

						Performance		
	Pass	**AC'**	**BC'**	**AC**	**BC**	**SF**	**DP**	**Speed**
CBC	false	NA	NA	NA	NA	NA	NA	16530
ELEPHANT	true	2	1	5	3	2.7	NA	22147
CURDIFF	true	2	3	5	3	1.6	1.6	6847
NEWDIFF	true	1	2	5	3	2.7	7.3	11580
NEWELF	true	0	1	5	3	8	NA	26820
NEWELFRED	true	0	1	1	2	4	NA	20860

8 Performance

We studied the performance of the optimized C versions of the ciphers/diffusers. For the diffusers we used the loop unrolling mechanism [20] and for the AES we used optimized Gladmann's implementation [21]. The results are listed in table 9 under column speed, note that the reported measurements are done on a PIV 3 GHz processor running on Windows Vista, where the programming environment was Microsoft VC++ 6. Here we reported the number of clock cycles needed by each algorithm, which is the minimum of 100 iterations to remove any initial overheads or cache misses. These results show that NEWDIFF is about 70% slower than CURRDIFF, NEWELF is about 20% slower than ELEPHANT and NEWELFRED is about 6% faster than ELEPHANT.

We define the Safety Factor (SF) with (3), which is the ratio between the total number of used diffusers' cycles over the minimum required. SF represents how safe is the current number of diffusers' cycles, under any circumstances this ratio should not be less than one. In (4), we defined the Diffusion Power (DP) , of a diffuser layer , to be the ratio between the number of bits updated per cycle (NC) over the total number of bits (TN) times SF. DP shows how fast the diffusion layer diffuse the plaintext/ciphertext. The values of SF and DP are reported in table 9. These values show that CURRDIFF possesses less SF and DP as NEWDIFF, and ELEPHANT possesses less SF that both NEWELF and NEWELFRED.

$$SF = (AC + BC) \div (AC' + BC') \tag{3}$$

$$DP = (NC) \div (TN) \times SF \tag{4}$$

From [11], suppose an attacker is attacking two identical hard drives, one encrypted with ELEPHANT and the other one encrypted with CBC. We are going to give the attacker the tweak key (K_{sec}), this means the attacker can now perform the diffusion layer for any plaintext. In other words, the diffuser layer becomes transparent to the attacker. All what is left now for the attacker is to attack the CBC layer, which is the same problem that he has when attacking the other hard drive (encrypted only using CBC). Although we helped the attacker significantly by providing him with the tweak key, he still has to attack the CBC layer. This shows that attacking ELEPHANT is not easier than attacking just CBC, and ELEPHANT is at least as secure as CBC. Note that the previous security proof is valid for any diffuser, that means NEWELF and NEWELFRED are also at least secure as CBC.

9 Discussion

CBC failed a lot of tests, as it is a narrow-block mode of operation [22], it possesses no avalanche effect at all and it is subjected to bit-flipping attack.

CURRDIFF is sensitive to repeated patterns, where its output can be distinguished from a random text and it is not so sensitive to the tweak change. We discovered also two cases where it will not change the input at all. It diffuses the plaintext/ciphertext slowly as at least five diffuser cycles, to pass **BD-Encryption** and **BD-Decryption** functions. It possesses also low SF and DP.

Due to the shortcomes of CURRDIFF, we designed NEWDIFF to replace it, the design of CURRDIFF was changed to update more bits each cycle and we added SBOX lookup operation to add non-linearity to the diffusers. NEWDIFF overcomes the shortcomes of CURRDIFF with good random profile, high SF and DP, but it is about 70% slower than CURRDIFF.

ELEPHANT is a wide-block mode of operation [22] that uses CURRDIFF together with CBC. Our analysis shows that it is superior than CBC, but the drawbacks of CURRDIFF can affect it, for example when CURRDIFF does not change the plaintext, ELEPHANT is reduced to CBC (although this may

happen with a very low probability, it is still a problem, as we can not restrict the plaintext). ELEPHANT possesses low SF.

NEWELF is a proposed variant of ELEPHANT, where we replaced CURRDIFF with NEWDIFF, it possesses good random profile and high SF, but it is about 20% slower than ELEPHANT.

NEWELFRED is a variant of NEWELF, where we reduced the number of diffuser cycles. Although it uses less number of cycles as NEWELF, it possesses a good random profile, with a higher SF than ELEPHANT and is about 6% faster than ELEPHANT.

10 Conclusion

We present a couple of statistical tests, that can be used to evaluate the behavior of ciphers that uses a diffuser followed by a mode of operation. We used these tests to study Windows Vista's disk encryption algorithm ELEPHANT. The algorithm provides better statistical and random behavior than CBC. Our study discovered some weaknesses in its diffusers, so we proposed new diffusers to replace them. Our proposed diffusers overcome the drawbacks of the current ones. We used the proposed diffusers to build a new variant of ELEPHANT called NEWELF, that possesses better properties than ELEPHANT with only 20% increase in its total running time. If performance is an issue, we proposed NEWELFRED that uses NEWDIFF with reduced number of cycles, it is faster than ELEPHANT and it possesses better properties than ELEPHANT.

References

[1] R. Anderson and E. Biham. Two practical and provable secure block ciphers: BEAR and LION. In *Dieter Gollmann, editor, Fast Software Encryption: Third International Workshop (FSE'96)*, 1996.

[2] S. Lucks. BEAST: A fast block cipher for arbitrary blocksizes. In *Patrick Horster, editor, Communications and Multimedia Security II, Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security*, 1996.

[3] P. Crowley. Mercy: a fast large block cipher for disk sector encryption. In *Bruce Schneier, editor, Fast Software Encryption: 7th International Workshop, FSE 2000*, 2001.

[4] S. Fluhrer. Cryptanalysis of the Mercy block cipher. In *Mitsuru Matsui, editor, Fast Software Encryption, 8th International Workshop, FSE 2001*, 2002.

[5] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback. Report on the Development of the Advanced Encryption Standard (AES). Technical report, 2000.

[6] A. Menezes, P. Van Oorschot., and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[7] P. Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. http://citeseer.ist.psu.edu/rogaway03efficient.html.

[8] D. McGrew and S. Fluhrer. The Extended Codebook (XCB) Mode of Operation. Cryptology ePrint Archive, Report 2004/278, 2004.

[9] S. Halevi and P. Rogaway. A tweakable enciphering mode. http://eprint.iacr.org/2003/148.
[10] S. Halevi and P. Rogaway. A parallelizable enciphering mode. http://eprint.iacr.org/2003/147.
[11] N. Ferguson. AES-CBC + Elephant diffuser : A Disk Encryption Algorithm for Windows Vista. http://download.microsoft.com/download/0/2/3/0238acaf-d3bf-4a6d-b3d6-0a0be4bbb36e/BitLockerCipher200608.pdf, 2006.
[12] Trusted Computing Group. TCG TPM Specification Version 1.2. www. trustedcomputinggroup.org.
[13] J. Daemen and V. Rijmen. AES Proposal: Rijndael. http://citeseer.ist.psu.edu/daemen98aes.html.
[14] J. Soto. Randomness Testing of the Advanced Encryption Standard Candidate Algorithms. http://citeseer.ist.psu.edu/article/soto99randomness.html, 1999.
[15] J. Soto and L. Bassham. Randomness Testing of the Advanced Encryption Standard Finalist Candidates. Computer Security Division,National Institute of Standards and Technology, 2000.
[16] M. El-Fotouh and K. Diepold. Statistical Testing for Disk Encryption Modes of Operations. Cryptology ePrint Archive, Report 2007/362, 2007.
[17] NIST statistical Suite. available at http://csrc.nist.gov/rng/rng2.html.
[18] C. Fruhwirth. New Methods in Hard Disk Encryption. http://clemens.endorphin.org/nmihde/nmihde-A4-ds.pdf, 2005.
[19] F. Webster and S. E. Tavares. On the design of s-boxes. In *Advances in Cryptology - Crypto 85. Lecture Notes in Computer Science. no. 218. H. C. Williams (editor)*, 1986.
[20] J. Davidson and S. Jinturkar. An aggressive approach to loop unrolling. Technical report, Department of Computer Science. University of Virginia. Charlottesville, 1995.
[21] B. Gladman. AES optimized C code. http://fp.gladman.plus.com/AES /index.htm, June 2006.
[22] IEEE P1619 homepage on Wikipedia. http://en.wikipedia.org/wiki/IEEE_P1619.