2D-Encryption Mode

AHMED A. BELAL*
MOEZ A. ABDEL-GAWAD**

March 2001

Abstract

In this paper, a new encryption mode, which we call the 2D-Encryption Mode, is presented. It has good security and practical properties. We first look at the type of problems it tries to solve, then describe the technique and its properties, and present a detailed mathematical analysis of its security, and finally discuss some practical issues related to its implementation.

I. INTRODUCTION

Imagine that two users, Alice and Bob, need to exchange some digital images. These images contain confidential data to both of them. They decide to use some block cipher to encrypt and decrypt the images, so that no one could see their contents. Accordingly, they agree on a key to use with the block cipher, so that they can safely store and exchange encrypted images.

All block cipher implementations would allow Alice and Bob to operate the block cipher in a certain mode of operation, usually the ECB or the CBC modes of operation [NBS80], [ANS83], and [ISO97].

In the ECB mode, encryption proceeds on the data where data is divided to blocks of size n bits (the size of the underlying block cipher), and each data block is encrypted independently of all other data blocks, using the same key. This results in a simple and fast encryption method. Because there are no data-dependencies between encrypted blocks, ECB mode also has the advantage that it could be implemented on parallel processors.

However, when the data encrypted becomes large the main problem of ECB becomes more apparent. The probability of encrypting the same data with the same key becomes larger, and this makes the job of an adversary easier. The adversary could build a code book of plaintext-ciphertext pairs (a known-plaintext attack), or at least she can deduce information from the encrypted image. A sample of such information deduction is shown in the Sample Images Appendix, in which the “1D” images were encrypted using DES in the ECB mode. The adversary could easily *see through* the encrypted images. She can deduce information related to objects inside them. The information she deduces could be the external outline of all objects in an image, the number of objects in that image, the kind and nature of these objects, or even the outline of each individual object. These might be the same kind of information that Alice and Bob want to hide. Note that the “2D” images, encrypted using the 2D-Encryption Mode, also show some information, but that is mainly due to the special value which the parameter BPR - discussed later - takes, and which is so chosen to let the Sample Images clearly illustrate the difference between 1D encryption and 2D encryption.

In the CBC mode of operation, it is required that the sender uses an Initialization Vector, *IV*. Encryption then proceeds by adding *IV* to the first data block and encrypting the result with the key. The result of the first encryption is then added to the following data block, and the result is fed to the encryption algorithm. Encryption proceeds in this manner until all data blocks are consumed [NBS80], [ANS83], and [ISO97]. CBC mode solves the main problem with the ECB mode, as it virtually decreases the probability that original repeated-data is encrypted with the same key twice, and so it prevents the adversary from building a code book of plaintext-ciphertext pairs. Moreover, CBC totally mixes-up the encrypted image, so that no information could be deduced from the encrypted image.

This added security of CBC does not come at no price. Although CBC is as fast (on one processor), and as simple as ECB, it does introduce a high level of data dependencies. This causes the CBC mode to lose the parallelism property of ECB. It could not be implemented on parallel processors, as each processor would have to wait for the results of its previous processor before proceeding in its operation. In addition, long sequences of data-dependencies, in general, cause problems when transferring encrypted data on computer networks.

The 2D-Encryption Mode, presented here, is a new encryption mode that is suited for all binary data, and, as a visual demonstration, we show how it could be applied in image encryption. The 2D-Encryption Mode tries to be a good compromise between the ECB and the CBC modes of operation, that gets as much good as possible from the two modes while trying to avoid as much of their problems. It is also more suited than ECB and CBC to some types of applications.

I.1 Related Work

Research is currently going on to study and enhance the properties of block ciphers’ modes of operation. In [BKR94], the first exhaustive analysis of the CBC MAC was presented. While in [BDJR97], the authors make a concrete security analysis of two popular modes: the CBC mode, and the counter (XOR) mode. They also present four notions of concrete security in their analysis. In [BI99] the authors try to simplify the analysis of encryption schemes, but their results were not exploited in our analysis, as the tools and concepts presented in [BDJR97] and [BKR94] were enough for our purposes.

In [RRY00] the authors stress the need for a mode that has a high degree of parallelism, and recommend against using modes with high levels of data dependencies. While in [Knu00] the author presents a detailed analysis of error-recovery and error-propagation properties of modes, and he gives some recommendations and, accordingly, presents the ABC mode.

In [LRW00] the authors present a review of the counter mode, while in [Jut00], [Rog00a], and [Rog00b] authors present other new modes of operation.

* Department of Computer Science and Automatic Control, Faculty of Engineering, Alexandria University, Egypt.
Email: belala@usa.net
** Informatics Research Institute, Mubarak City for Scientific Research and Technological Applications, Alexandria, Egypt. Email: moezaabdelgawad@hotmail.com

§ A longer version of this document is available at http://www.geocities.com/moezaabdelgawad/Research/2DEM.ps.zip
It is submitted to NIST while it revises and updates standard encryption modes. See http://www.nist.gov/modes

II. Description of The 2D-Encryption Mode

The 2D-Encryption Mode (2DEM, for short) naturally extends normal 1D encryption to the 2D case. It takes as its input an image, or more abstractly a binary stream of data. It proceeds by first performing what is basically an ECB encryption phase on (the rows of) the image, treated as one continuous successive stream of blocks, then it performs another ECB encryption phase on the columns of the resulting image, also treating them as one successive continuous stream of blocks.

An algorithmic description of the mode, in pseudocode, is given below. The pseudocode implements the 2DEM mode on any 2D data (later, it is shown how 2DEM is applied to 1D data). The pseudocode assumes using DES as the underlying block cipher, and the data is assumed to be divided to 8 bit blocks (bytes). Data is supplied in a 2-dimensional array of bytes called "Data", and its byte values are accessed by supplying a row number and a column number. The number of rows in the array Data is numrows, while the number of columns is numcols.

```
(* Phase 1 : Row-Encryption *)
for row = 0 to numrows - 1 do
  for col = 0 to numcols - 1, col += 8 do
    DES(Data(row, col)..Data(row, col+7))

(* Phase 2 : Column-Encryption *)
for col = 0 to numcols - 1 do
  for row = 0 to numrows - 1, row += 8 do
    DES(Data(row, col)..Data(row+7, col))
```

The ".." notation is used to denote the (eight) data values between the two boundary data bytes.

The algorithm assumes numcols and numrows are multiples of 8, which is the result of dividing n by 8 (the size of a byte). For a 128-bit block cipher (such as AES) numcols and numrows need to be multiples of 16 (=128/8). The operation DES, in the algorithm, encrypts the 64-bits of data supplied to it, using the DES algorithm and the key that the users have agreed upon, and the result of the encryption is returned to the same locations which the input used to occupy in the array "Data".

For a general block cipher F, of size n bits (where n is assumed to be a multiple of 8), the value $n/8$ would be called NSB (number of sub-blocks), and the algorithm would be as follows:

```
(* Phase 1 : Row-Encryption *)
for row = 0 to numrows - 1 do
  for col = 0 to numcols - 1, col += NSB do
    F(Data(row, col)..Data(row, col+NSB-1))

(* Phase 2 : Column-Encryption *)
for col = 0 to numcols - 1 do
  for row = 0 to numrows - 1, row += NSB do
    F(Data(row, col)..Data(row+NSB-1, col))
```

Later, we would map numcols (the number of bytes in a row) to the BPR (blocks per row) parameter. BPR would be a randomly generated integer that defines the width of a row of data. Thus, BPR defines the extent of interleaving of data, and determines which data bytes are to be fed to the underlying block cipher, F, as the columns of the 2D image (or the array "Data" in the pseudocode above).

II.1 2D and 1D Examples

We demonstrate how 2DEM operates by presenting an example. Then we show how 2DEM is applied to 1D data.

Assume we have a 16 × 16 pixels image. Each pixel has a gray-scale color in the range 0..255 (and thus occupies eight bits), so eight pixels (in a row or in a column) would form one 64-bit DES data block. For simplicity, assume the color values in the image are as follows (in hexadecimal notation):

```
00, 01, 02, 03, 04, ... , 0B, 0C, 0D, 0E, 0F
10, 11,             ...             , 1E, 1F
20, 21,             ...             , 2E, 2F
 .                   .                     .
 .                   .                     .
 .                   .                     .
D0, D1,             ...             , DE, DF
E0, E1,             ...             , EE, EF
F0, F1, F2, F3, F4, ... , FB, FC, FD, FE, FF
```

Moreover, the key used in DES encryption is k = 1234567890ABCDEF.

2DEM proceeds by encrypting the 64-bit block 0001020304050607, the first eight pixels in the first row of the image, using DES with the key k. Then it encrypts the block 08090A0B0C0D0E0F. The technique then follows to encrypt 1011121314151617, which is the first block of the second row, and so on it continues, until it reaches the final *row* block in the image, i.e., the block F8F9FAFBFCFDFEFF. The intermediate image (equivalent to an ECB encryption of the rows of the image) would be as follows:

```
70, 7F, 12, 2A, 2B, ... , CF, 7C, 6A, 67, 7B
67, C7,             ...             , DF, 7A
CA, 2F,             ...             , CE, 48
0E, 6C,             ...             , 4F, 98
4F, A6,             ...             , C7, 3D
86, 35,             ...             , 78, B8
4D, 7B,             ...             , 38, 61
15, A4,             ...             , 7F, 8B
44, 20,             ...             , 8F, 46
C8, E4,             ...             , 37, 3A
04, 8B,             ...             , 35, 19
3F, 18,             ...             , 3B, D1
22, 19,             ...             , 87, 33
C4, 41,             ...             , 95, C0
46, E0,             ...             , ED, AA
64, 0A, 63, F7, 48, ... , 28, CE, B2, D5, 7B
```

Then, 2DEM enters its second phase, the column-encryption phase. It encrypts the 64-bit *encrypted* column block 7067CA0E4F864D15 (see intermediate image above), then encrypts the second column block 7FC72F6CA6357BA4. It then finishes encrypting the first eight rows by encrypting the column block 7B7A48983DB8618B. 2DEM, then, proceeds to encrypt the block 44C8043F22C44664. It then continues encryption of the columns of the last eight rows of the image, till it reaches the final block, 463A19D133C0AA7B. The final 2DEM encrypted image, then, would be:

```
14, 36, F4, 72, E1, ... , 82, 50, 07, 39, E0
08, 5A,             ...             , 36, 86
B3, 04,             ...             , 73, B9
26, 0A,             ...             , C2, DE
AB, A4,             ...             , 78, 6F
7C, 4B,             ...             , 0C, A4
D9, C1,             ...             , 26, 65
68, 51,             ...             , A2, 16
E9, 38,             ...             , 02, D2
F4, B6,             ...             , 24, E5
00, F4,             ...             , 0C, 97
92, 0D,             ...             , D0, 99
0C, CA,             ...             , 9E, E6
A8, 20,             ...             , B2, 5A
86, E2,             ...             , BE, CB
B2, BE, C1, 34, 43, ... , 31, F0, C8, BA, 62
```

Although the 2DEM mode seems - because it is described above in 2D terms - to be more suited for encrypting images, and data of 2D nature (such as relational database tables, or spreadsheet tables), 2DEM could be applied to other forms of data such as text-files, word-processor documents, and other data of 1D nature. 2D data have natural row and column boundaries, and the columns of the data are just as easily accessible as its rows. For 1D data, this could be simulated, by defining a row width and making appropriate coding of data access. (2DEM is designed to make software and hardware implementations efficient for both 1D and 2D data). By defining a row width (`numcols` or BPR) and defining data separated apart by "row width" to be in the same column, 2DEM could operate normally on the new artificial rows and columns.

An example of how 2DEM would be applied to 1D data, is presented below. This extra example would be beneficial also in fostering a better understanding and interpretation of the security and error-propagation analysis of 2DEM which, for compatibility and consistency with all research work in this field, is presented in 1D terms.

We use the term "sub-block" to refer to a block of bits of length n, which is input to the underlying block cipher; while the two terms "2D block" and "$n^2/8$-bits block" are used synonymously to refer to the block of bits of size $n^2/8$ bits that is supplied to 2DEM. In 2DEM, this 2D block is subdivided to NSB sub-blocks that are fed to the underlying block cipher F, where in the first phase they are fed as rows of the plaintext, while in the second phase they are fed as columns of the *encrypted* rows.

Using DES as the underlying block cipher, our example assumes a message of size 384 bytes. So, the size of the message, in bits, is $M = 384 \times 8 = 3072$ bits $= 6 \times 512$, i.e., six 2D blocks.

Furthermore, assume that the message is composed of the following byte values (in hexadecimal notation):

```
00, 01, 02, 03, 04, ... , FD, FE, FF, 7F, 7E,
7D, 7C, 7B, ... , 02, 01, 00
```

This 1D message could be encrypted using 2DEM as follows. The message is divided to six blocks of size $n^2/8 = 512$ bits each. The rows of the 2D blocks that are input to 2DEM, are then selected from these six blocks according to the parameter BPR. Notice that BPR has a minimum value of 1, while its maximum possible value is $M/(n{\cdot}\text{NSB}) = M/(n^2/8)$.

Each row in a 2D block is of length n bits (i.e., each row is a sub-block), and these rows are selected such that consecutive rows are BPR sub-blocks apart in the original message. This is equivalent to interleaving the rows of BPR consecutive 2D blocks that are chosen sequentially from all the $n^2/8$-bit blocks of the message.

To summarize, this implies that the plaintext (a 1D stream of bytes) is treated as a 2D array of bytes whose `numcols` = BPR·NSB.

For the message given above the possible values for BPR would be 1, 2, 3, 4, 5, and 6. BPR is randomly generated from this set of possible values. If BPR takes the value 1, then the six 2D blocks that get encrypted would be as follows:

```
00, 01, ... , 06, 07
08, 09, ... , 0E, OF
10, 11, ... , 16, 17
.       ...        .
.       ...        .
.       ...        .
30, 31, ... , 36, 37
38, 39, ... , 3E, 3F

40, 41, ... , 46, 47
48, 49, ... , 4E, 4F
50, 51, ... , 56, 57
.       ...        .
.       ...        .
.       ...        .
70, 71, ... , 76, 77
78, 79, ... , 7E, 7F

80, 81, ... , 86, 87
88, 89, ... , 8E, 8F
90, 91, ... , 96, 97
.       ...        .
.       ...        .
.       ...        .
B0, B1, ... , B6, B7
B8, B9, ... , BE, BF

C0, C1, ... , C6, C7
C8, C9, ... , CE, CF
D0, D1, ... , D6, D7
.       ...        .
.       ...        .
.       ...        .
F0, F1, ... , F6, F7
F8, F9, ... , FE, FF

7F, 7E, ... , 79, 78
77, 76, ... , 71, 70
6F, 6E, ... , 69, 68
.       ...        .
.       ...        .
.       ...        .
4F, 4E, ... , 49, 48
47, 46, ... , 41, 40

3F, 3E, ... , 39, 38
37, 36, ... , 31, 30
2F, 2E, ... , 29, 28
.       ...        .
.       ...        .
.       ...        .
0F, 0E, ... , 09, 08
07, 06, ... , 01, 01
```

while if BPR takes the value 2 then the six 2D blocks that get encrypted would be as follows:

00, 01, ... , 06, 07 10, 11, ... , 16, 17 20, 21, ... , 26, 27 . ... 60, 61, ... , 66, 67 70, 71, ... , 76, 77	08, 09, ... , 0E, OF 18, 19, ... , 1E, 1F 28, 29, ... , 2E, 2F . ... 68, 69, ... , 6E, 6F 78, 79, ... , 7E, 7F
80, 81, ... , 86, 87 90, 91, ... , 96, 97 A0, A1, ... , A6, A7 . ... E0, E1, ... , E6, E7 F0, F1, ... , F6, F7	88, 89, ... , 8E, 8F 98, 99, ... , 9E, 9F A8, A9, ... , AE, AF . ... E8, E9, ... , EE, EF F8, F9, ... , FE, EF
7F, 7E, ... , 79, 78 6F, 6E, ... , 69, 68 5F, 5E, ... , 59, 58 . ... 1F, 1E, ... , 19, 18 0F, 0E, ... , 09, 08	77, 76, ... , 71, 70 67, 66, ... , 61, 60 57, 56, ... , 51, 50 . ... 17, 16, ... , 11, 10 07, 06, ... , 01, 00

and so on for the other values of BPR. Notice that, the bytes of the encrypted 2D block would occupy the same positions, in the ciphertext array, as their corresponding plaintext did in the plaintext array. Also, note that for BPR = 4 or 5 we would have shortage of 2D blocks. So either the message is padded with enough blocks, or the blocks of the "last row of 2D blocks" could be encrypted without interleaving, or encrypted with an interleaving factor equal to the number of remaining blocks (less than BPR), or ciphertext-stealing (or some other alternative to padding) [Sch95] could be used.

In the next section we present an evaluation of 2DEM, while mathematical analysis of its security, error-recovery and error-propagation properties is presented in section IV, and finally section V discusses some practical issues related to its design and implementation.

III. Evaluation of The 2D-Encryption Mode

III.1 Evaluation of the Security of the 2D-Encryption Mode

With respect to security, the 2DEM mode has the following properties:

i. The probability of encrypting the same data block using the same key becomes much lower than that of ECB, that is because the size of the data block that needs to be repeated has increased from 64 bits to 64 × 8 = 512 bits for DES as the underlying block cipher, or has increased from 128 bits to 128 × 16 = 2048 bits for AES as the underlying block cipher. The memory requirements for mounting a successful birthday attack, thus, have increased approximately from 2^{32} to 2^{256} for 2DEM using DES, and approximately from 2^{64} to 2^{1024} for 2DEM using AES.

ii. 2DEM, due to the extra encryption phase it performs, hides the statistical information of the original message more than the ECB mode does. The row-encryption phase (the first phase) randomizes the input to the column-encryption phase (the second phase). The effect of this is clear in the statistics and histograms accompanying each of the Sample Images in the Appendix. These statistics and histograms show that the number of colors, in encrypted images, increases from 1D encryption (i.e., using ECB) to 2D encryption (i.e., using 2DEM). In addition, the maximum repetition for a single color decreases; the minimum repetition for a single color increases; and the variance between color repetitions decreases and the average color of the whole encrypted image approaches the middle of the color scale. (a gray-scale of colors from 0 to 255 is used, so the midpoint of the scale is 127.5).

iii. By vertically encrypting data that lie in the same column, 2DEM encrypts data blocks that are far apart by BPR sub-blocks. This lets 2DEM be more resistive to the "block replay" attack, which is a serious security problem with the ECB mode [Sch95], and has a counterpart in the CBC mode [Knu00]. Because it defines and uses a row width (numcols, or alternatively BPR), 2DEM correlates data blocks that are not logically related to each other, which is the main feature exploited by the block replay attack in ECB mode. This is even more effective when the width of rows is chosen at random, or could be chosen and changed easily by the users.

It is to be noted that we discuss and evaluate in this paper "bare" 2D encryption using 2DEM, and we mainly compare it with "bare" 1D encryption using ECB, i.e., with no *IV*s (as in the CBC mode) and no *nonces* or counters (as in the Counter mode). These could be equally and easily added to both 1D and 2D encryption modes, to add extra security. For the counters, the positional counter in 1D encryption could be split to two counters in 2D encryption: a row positional counter and a column positional counter. These are extra suggestions, but are not investigated furthermore here, and are not included in the specification of the 2D-Encryption Mode, as we feel it is appropriately secure.

III.2 Evaluation of the Performance of the 2D-Encryption Mode

With respect to performance, the 2DEM mode has the following properties:

i. Speed: Implemented on one processor, 2DEM takes just double the time the ECB mode takes. While this is more than the time required for the CBC mode, 2DEM regains the parallelizability property of ECB. Each block of 8 × 64 bits (or 16 × 128 bits) could be encrypted independently from other blocks, so 2DEM could be implemented on parallel processors, and thus substantially reduce the encryption time. Also the 8-byte (or 16-byte) rows of each 2D block, in the first phase, could be encrypted in parallel independently of each other; while in the second phase the 8-byte (or 16-byte) columns of the block could be encrypted in parallel, also independently of each others. That means that when enough processors are available, the time to encrypt the whole 2D block (and, accordingly, also the whole message) could be just double the time required to encrypt one sub-block. That is because the length of the longest critical path in the 2D-Encryption Mode is two, and in fact all critical paths have a length of two.

ii. Memory requirements: if 2DEM is implemented on parallel processors, each processor requires just its 8 × 64 bits (or 16 × 128 bits) block. When implemented on a single processor then just only eight (or sixteen) rows of the original data need to be held in memory. This could be reduced, with some handling and processing, to just the 8 × 64 bits block (or 16 × 128 bits) if the whole data is available on permanent storage, and is not, for example, downloaded from a network. Thus, 2DEM virtually adds no memory requirements beside that required for implementing the underlying block cipher (for its tables and S-boxes, for example), and the memory required for keeping the parameter BPR (an integer value).

iii. Error-Propagation and Error-Recovery: 2DEM limits the effect of bit errors only to the encrypted/decrypted 8 × 64 bits (or 16 × 128 bits) block of data. Extra analysis of the error-recovery and error-propagation properties of 2DEM is presented in the next section.

IV. Mathematical Analysis

In this section, we present an analysis of the security and error-propagation properties of the 2D-Encryption Mode.

IV.1 Security Analysis of the 2D-Encryption Mode

When using a block cipher with block size *n* bits, it is shown in the description of 2DEM that it works on 2D blocks of data of size NSB rows by NSB columns, and each element has a size n / NSB bits. This implies that the size of the 2D blocks is NSB · NSB · (n / NSB) = n · NSB bits. For our choice of NSB = n / 8, the size of the 2D blocks is n · (n / 8) = $n^2/8$ bits (See § V for the rationale behind choosing NSB = $n/8$). The row-encryption and column-encryption phases of 2DEM are then repeated for each block of data, where the data of the columns are input to the block cipher according to the number of blocks per row, BPR.

To start our analysis, let us denote the encryption of the 2D block of size $n^2/8$ by "2D Encryption", and denote the usage of the parameter BPR with "2D Encryption" as "2D Mode" (BPR determines which data are fed together to "2D Encryption" as a 2D block).

Our proof for the security of the 2D-Encryption Mode proceeds in four consecutive steps. The proof is divided into two parts, and each part is additionally divided to two steps.

The first part proves the security of "2D Encryption" by proving that it produces a wide PRP family from a small PRP family (e.g., DES or AES), and the second part proves the security of "2D Mode", which using BPR, allows

"2D Encryption" to be used on messages of arbitrary length M.

The two steps of proving security of "2D Encryption", and the two steps of proving security of "2D Mode" are similar. The first step in each part proves security assuming the underlying block cipher is a random permutation family, while the second step extends this to the case where it is assumed that the underlying block cipher is a pseudorandom permutation (PRP) family.

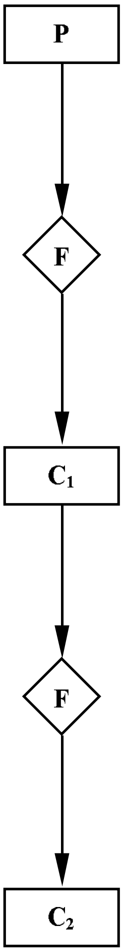
In the analysis of "2D Mode", it is also assumed that the adversary knows the value of the parameter BPR, and that she is performing a known-plaintext attack. These assumptions, together with an assumption made when analyzing the security of "2D Encryption" imply that, in fact, the 2D-Encryption Mode may have better security limits than those proven here. (We target, here, to make a worst-case analysis).

IV.1.1 Security of "2D Encryption"

In this part, we prove in two steps that "2D Encryption" produces a wide pseudorandom permutation (PRP) family from a small PRP family (e.g., DES or AES).

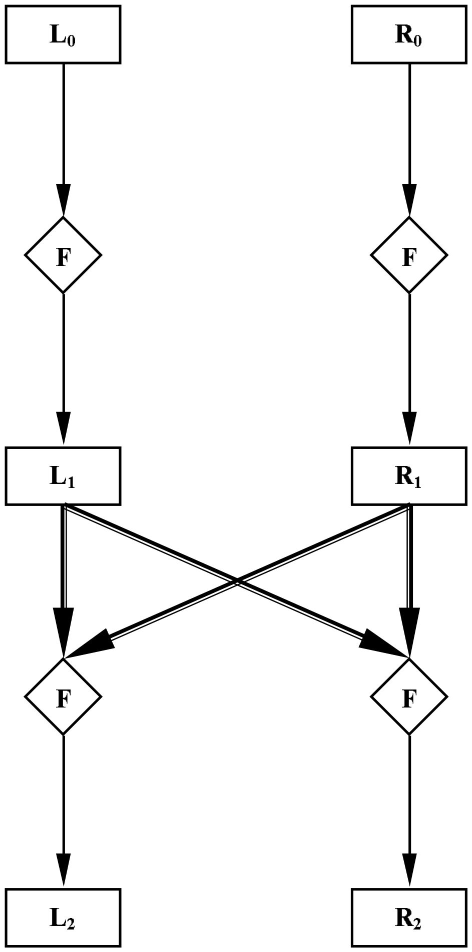
Proving "2D Encryption" secure in the "random permutation family" model means "2D Encryption" provides a random permutation family if the underlying block cipher is a random permutation family. The proof is direct and straightforward. Next, this is extended to the case where the underlying block cipher is a PRP family, where "2D Encryption" is proved to produce a PRP family if the underlying block cipher is a PRP family. Together, both facts imply that "2D Encryption", while extending the size of the input/output blocks, preserves the security properties of the underlying block cipher, and that is at the cost of an extra encryption per sub-block. This loss in performance is regained by exploiting the high degree of parallelism inherit in the design of the algorithm.

The following two figures are used in our proofs. They show simplified versions of "2D Encryption" and they help demonstrate how the proof goes on. It takes a little bit of imagination to see that figures (1) and (2) represent the "2D Encryption" part in 2DEM with NSB = 1, and NSB = 2, respectively.



$C_2 = F(\ F(\ P\ )\ )$

Figure (1)
2D Encryption with *n* = 8 (or NSB = 1)



$L_2 = F(\ F(\ L_0\ )|_L \ ||\ F(\ R_0)|_L\ )$ $R_2 = F(\ F(\ L_0\ )|_R \ ||\ F(\ R_0)|_R\ )$

Figure (2)
2D Encryption with *n* = 16 (or NSB = 2)

In the equations above, "||" represents the concatenation operator, while "$|_L$" represents the permutation that outputs the left half of its input, and "$|_R$" outputs the right half.

Having the notation necessary for our four-pronged proof laid out, here is the first step of it.

IV.1.1.A "2D Encryption" is secure in the "random permutation family" model

Intuitively, the proof says that, since we assume the input is random and we are using a random permutation family and use (in a randomness-preserving way) operations which preserve the randomness property then we have a random output.

Formally, the input block of bits to "2D Encryption" (of size $n^2/8$) is assumed to be a random variable. The following three operations are used within "2D Encryption": (see figures (1) and (2))

a. Compression permutations, which are permutations that output a finite subset of the bits of their input. These permutations are used between the row-encryption phase and the column-encryption phase. In the figures, the permutations are represented by the arrows between the top encryptions (row-encryption phase) and the bottom encryptions (column-encryption phase). Notice that the permutations select *disjoint* subsets of their inputs.

b. The concatenation operation, which outputs the bit concatenation of its two (left and right) inputs.

c. Computing F (the underlying block cipher) using the key *k*.

Each of these three operations produces a random variable if its inputs are random variables (and for F, its output is also assumed to be totally independent from its input. That is part of the definition of a random permutation family).

Let us, then, analyze the case where NSB = 2. From figure (2), the output equals $L_2 \| R_2$ (the concatenation of L_2 and R_2) where $L_2 = F(\ F(L_0)|_L \ \|\ F(R_0)|_L\ )$ and
$R_2 = F(\ F(L_0)|_R \ \|\ F(R_0)|_R\ )$.

L_2 is the output of F computed on the concatenation of the left parts of the outputs of F, computed for L_0, R_0. Since we assume the input is a random variable, then L_0, R_0 are totally independent and each is a random variable of size 8 bits. So the two outputs of F when applied to L_0, R_0 are two independent random variables. In figure (2), these are L_1, R_1.

Accordingly, the left parts of L_1, R_1 are also two independent random variables (from the properties of the compression permutation implied here), and they are also independent of the right parts of L_1, R_1 which are used in computing R_2. (From here comes the importance that the compression permutations select *disjoint* subsets of bits).

Using the same argument for R_2, the same conclusions could be reached; and so we have L_2, R_2 as the computations of F for the two independent random variables v_l, v_r where $v_l = F(L_0)|_L \| F(R_0)|_L$ and $v_r = F(L_0)|_R \| F(R_0)|_R$, and so (again, because F is a random permutation family) we have L_2, R_2 as two independent random variables, and so the output (which is their concatenation) is a random variable which is totally independent of v_l, v_r and so independent of L_0, R_0.

This proves that "2D Encryption" produces a random permutation family assuming the underlying block cipher is a random permutation family. (Q.E.D)

A similar argument applies for the trivial case where NSB = 1, in figure (1); and the argument can be extended to the practical cases where NSB = 8 (for DES) or NSB = 16, 24 or 32 (for AES with 128, 192 or 256 data bits, respectively). (In the longer version of this paper a diagram for the case of NSB = 4 is presented).

Another proof considers the probability of obtaining some output value assuming a certain input value. The probability is proved (using an argument similar to the one used above) to be $2^{-n\cdot NSB} = 2^{-\frac{n^2}{8}}$ for a certain input value and all output values, and vice versa.

It is appropriate to notice at this point that the security of "2D Encryption" increases as n·NSB increases (the probability of obtaining an output value decreases when the size of output increases), so it is clear that the larger the value of NSB the more the security gained (the maximum possible value for NSB is n), but the value of NSB had to be sensibly chosen taking in consideration security, performance and flexibility issues (See § V: 2D-Encryption Mode and Some Practical Issues, for further discussion about choosing the value of NSB).

IV.1.1.B Extending "2D Encryption" to the real world of PRPs

This step relates the security of "2D Encryption" to the security of the underlying block cipher (PRP). It is assumed that the adversary *A* mounts a chosen-plaintext attack against "2D Encryption", and we relate her advantage in distinguishing it from a random permutation family over $n^2/8$ bits to her advantage in distinguishing F (the underlying block cipher) from a random permutation family over n bits.

For an adversary *A* to mount a chosen-plaintext attack against "2D Encryption", she must be provided with an oracle *O* that supplies her with the ciphertext corresponding to a plaintext 2D block that she chooses, encrypted with the users' key *k* (which is unknown to her).

We also assume that the adversary knows the intermediate results of invocating F, i.e., the output of the row-encryption phase (these are L_1, R_1 in figure (2)). This assumption makes the analysis untight, as an oracle of F is not sure to be available to the adversary *A* (she is only guaranteed access to the "2D Encryption" oracle), but this assumption simplifies the analysis, and provides an upper-bound on the success probability of the adversary in attacking F.

The resources available to the adversary are the running time *t* (which includes the space for her program), and the number of queries *q* she makes to the "2D Encryption" oracle. The parameter we are interested in calculating is ∈, the adversary's advantage over simple guessing. For a permutation family to be a "random permutation family" ∈ must be zero irrespective of *t* and *q*. For a permutation family to be a "pseudorandom permutation family" ∈ must be negligible (See [NR97] for more details).

Assume that the adversary *A* can (*t*, *q*, ∈)-break "2D Encryption". This means that *A* runs in time *t*, makes *q* queries to her "2D Encryption" oracle, and succeeds with advantage ∈ in distinguishing the output of "2D Encryption" from a random permutation over $n^2/8$ bits. Our results specify *t'*, *q'*, ∈' as functions of *t*, *q*, ∈, *n*, such that there exists an adversary *A'* (a simple modification of *A*) that (*t'*, *q'*, ∈')-breaks the underlying block cipher, F. Current knowledge gives us values *t'*, *q'*, ∈' for which it seems safe to rule out (*t'*, *q'*, ∈')-breaks for typical F. From this we can derive values for which (*t*, *q*, ∈)-breaks of "2D Encryption" are effectively ruled out. Thus, the security of "2D Encryption" is reduced to that of F in a constructive and useful way.

Proof:

Intuitively, the previous step in the proof, says that "2D Encryption" assuming F is a random permutation family is secure. If "2D Encryption" were not secure, this would mean F is not good as a PRP family.

Formally, a construction argument is used to prove that "2D Encryption" is a PRP provided F is a PRP. The construction provides the adversary A with a means to break F with advantage $\in'$, if she could break "2D Encryption" with advantage $\in$.

Again, we analyze the case where NSB = 2. The results are then easily extended to other cases.

From figure (2), we see that the output $L_2 \| R_2$ would be seen as a "random permutation" of the input $L_0 \| R_0$, if and only if, each of the four invocations of F acts as a "random permutation". This implies that $L_2 \| R_2$ does not seem random to the adversary A (with probability $\in$) when the input/output behavior of at least one of the four invocations of F does not seem random.

This implies that the probability of success of A in attacking "2D Encryption" equals the sum of the success probabilities for all invocations of F. Since the latter probabilities are all, by definition, equal to $\in'$, the adversary (using one query to the "2D Encryption" oracle, O) has $\in = 4\cdot\in'$, and since each query to "2D Encryption" maps to four queries to F, then if A issues q queries to the oracle O, we have

$$q = q'/4,$$

$$\text{and } \in = 4\cdot q\cdot\in'.$$

For the general case, we would have

$$q = \frac{q'}{2\cdot NSB} = \frac{4\cdot q'}{n},$$

$$\text{and } \in = 2\cdot\text{NSB}\cdot q\cdot\in' = \frac{q\cdot n}{4}\cdot\in',$$

and so

$$q' = 2\cdot\text{NSB}\cdot q = \frac{q\cdot n}{4},$$

$$\text{and } \in' = \frac{\in}{2\cdot NSB\cdot q} = \frac{4}{q\cdot n}\cdot\in.$$

Also, the time used to break F is the same used in querying the "2D Encryption" oracle O, so $t' = t$.

Thus, the given values of t', q', $\in'$, provide us with a construction of an adversary A' that $(t', q', \in')$-breaks F.

(Q.E.D)

(A more optimistic analysis, that assumes intermediate results are unknown, gives $\in = q\cdot\text{NSB}^2\cdot\in'^2$. This more optimistic $\in$ is smaller than the one above since $\in'$, by definition, is very small (negligible) and so is smaller than 2/NSB ($=16/n$) which is constant for a certain block cipher. For DES, this constant is equal to 1/4, and for 128-bit AES it is equal to 1/8).

IV.1.2 Security of "2D Mode"

The second part in the security analysis of the 2D-Encryption Mode involves the analysis of "2D Mode". That is the usage of the parameter BPR with "2D Encryption", which allows "2D Encryption" to be used on arbitrary length messages. "2D Mode" is built on the structure and design of "2D Encryption". Using BPR, "2D Mode" defines the location of the data in the message that will constitute the 2D blocks of "2D Encryption" (of size $n^2/8$ bits each).

If BPR is known or is calculated according to some fixed rule, then "2D Mode" adds no security to "2D Encryption" ("2D Encryption" has its own large-enough security parameters). That is because, in this case, "2D Mode" would be the same as applying ECB, but using large blocks of size $n^2/8$, and so the likelihood of a collision (successful birthday attack) is greatly reduced. (See the third step of the proof, below. It presents exact equations). But a random unknown BPR adds much to the security of messages, as it aims at preventing practical attacks other than the birthday attack. This point is further discussed, separately, after the fourth step of the proof.

Like the proof for "2D Encryption", the proof of security of "2D Mode" proceeds in two steps. In both steps, it is assumed that the adversary knows the value of BPR (and, w.l.o.g., BPR = 1), and that she performs a known-plaintext attack. These two assumptions make the security analysis of "2D Mode" be the same as that of ECB, and thus the security parameters would be easily deduced.

ECB could be analyzed only assuming a known-plaintext attack. Using a chosen-plaintext attack, the adversary A could easily produce a collision in ECB mode, by *choosing* two equal input blocks. "2D Mode" prevents this attack on ECB, by using the random parameter BPR, and making the cost of confirming a guess of BPR very large (quadratically exponential).

IV.1.2.A "2D Mode" is secure in the "random permutation family" model

We prove here that if A is an adversary performing a known-plaintext attack against "2D Mode", using q plaintext-ciphertext pairs, totaling at most μ bits, then the advantage $\mathtt{Adv}$ of A in distinguishing the input-output behavior of "2D Mode" from a "random permutation" on the plaintext is upper-bounded by the following equation:

$$\mathtt{Adv} \le 1-(1-2^{-\frac{n^2}{8}})^{\frac{q(q-1)}{2}}$$

Proof:

[Knu00] states the fact that for an l-bit block cipher used in ECB mode, where the plaintext blocks are assumed to be chosen at random from a uniform distribution, if s blocks are encrypted under the same key then information is leaked about some plaintext blocks with probability $p_s = 1 - (1 - 2^{-l})^{s(s-1)/2}$, and for $s \le 2^{(l+1)/2}$ we have $p_s \le 0.632$.

For ECB, this is the birthday attack, where the adversary given two equal ciphertext blocks knows that the corresponding plaintext blocks are equal. Thus, the given equation places an upper bound on the success probability of the adversary in mounting a birthday attack against ECB.

Mapping this to the analysis of "2D Mode" makes us replace l, in the equation of p_s by $n^2/8$, and replace s by q, and thus $p_q = 1-(1-2^{-\frac{n^2}{8}})^{\frac{q(q-1)}{2}}$, and for a success probability $p_q \ge 0.632$, the adversary A requires $q \ge 2^{\frac{\frac{n^2}{8}+1}{2}}$ queries, each of size $n^2/8$ bits.

This shows that "2D Encryption" has *quadratically exponential* increased the amount of ciphertext needed to mount a birthday attack against "2D Mode".

Since we are assuming the underlying block cipher is a "random permutation family", the adversary realizes a non-random output of "2D Mode" only if she sees two equal 2D blocks of ciphertext, so we have for q queries ($q \leq 2^{\frac{\frac{n^2}{8}+1}{2}}$), and $\mu = q \cdot (n^2/8)$ bits, the advantage of A = Adv where $Adv = p_q \leq 1-(1-2^{-\frac{n^2}{8}})^{\frac{q(q-1)}{2}}$ (Q.E.D)

The last step in our proof, extends this to the "real-world" case of PRPs.

IV.1.2.B Extending "2D Mode" to the real world of PRPs

This is a standard proof (See [BDJR97]). It is omitted for space considerations. It follows a similar line to that given for the second step of "2D Encryption" when extending it to the real world of PRPs. It intuitively says that if "2D Mode" were not secure this would mean that the underlying block cipher ("2D Encryption" in the case of "2D Mode") is not good as a PRP family.

IV.1.2.C The Role of BPR in "2D Mode" and the 2D-Encryption Mode

As mentioned earlier, the parameter BPR aims at preventing attacks other than the birthday attack. These are practical attacks such as the "block replay" attack [Sch95], and other ciphertext manipulation attacks. BPR presents a factor of uncertainty facing the adversary, since it would take some random value in the range 1..M/(n^2/8). Moreover, to confirm a guess the adversary has to mount an expensive birthday attack.

Usually a small set of possible values for BPR is enough to totally confuse an adversary and prevent her from performing ciphertext manipulation attacks. If the adversary has a guess for the value of BPR, she has to perform a very expensive birthday attack to be certain of her guess or to renounce it. Using her guess of BPR and performing a known-plaintext attack, the adversary A has to find two equal plaintext 2D blocks (of size n^2/8 bits), and if the two corresponding ciphertext 2D blocks are equal then her guess is correct, else she renounces it. Thus, the main purpose of the unknown BPR is to prohibit the adversary from trying to know the effect (on the plaintext) of any changes she makes to the ciphertext.

Since the range of possible values of BPR is 1 .. M/(n^2/8), we notice a nice behavior of BPR. The range of possible values for BPR increases when M increases. So for larger messages, where plaintext blocks are more probable to repeat, the range of possible values of BPR increases and thus an adversary faces more difficulty in guessing BPR.

The usage of the parameter BPR is assumed to be as follows: it is generated randomly from a set of possible values, then it is used in encrypting the message using 2DEM, and finally the parameter is encrypted itself (using the users' key k) and sent with the ciphertext. The BPR could be encrypted using 1D encryption, and, because it is not part of the original message, mounting an attack on it would not pose a threat to the original message. If maximum security is required, the BPR could be encrypted in a separate 2D block using 2DEM, but that is not strictly required in 2DEM. The only necessary requirement regarding the parameter BPR, is that Bob (the receiver) knows BPR *before* he starts decrypting the message. This is the same condition required for the *IV* or *nonce* values used in the CBC or counter modes, but although these could be public [Sch95], the parameter BPR adds security to the 2DEM mode, and so should be encrypted.

Also, note that for a message (or data) of length M bits, the possible values of BPR (1..M/(n^2/8)) are allowed to be further limited by some other practical considerations, since BPR is a random value that is under the total control of the users, Alice and Bob. So if they gain some practical advantage by excluding some (context-specific) values of BPR, or further limit themselves to a subset of all possible values of BPR, they are free to do so. That is possible as long as there remains a set of possible values for BPR, and the value of BPR used is chosen randomly from this set, and is totally independent from the message encrypted. This freedom is possible because 2DEM is proved to be secure using the assumption that BPR is known, and w.l.o.g. equal to 1.

IV.2 Error-Recovery and Error-Propagation Properties of the 2D-Encryption Mode

[Knu00] defines the *error-recovery* property of an encryption mode as: the property that an error in the ith ciphertext block is inherited by only a few plaintext blocks after which the mode resynchronizes. It also defines *error-propagation* as: the property that an error in the ith ciphertext block is inherited by the ith and all subsequent plaintext blocks (sometimes error-recovery is called finite error-propagation, while error-propagation is called infinite error-propagation). The author advocates infinite error-propagation. Mainly he argues that the more the error-propagation in a mode the more secure the mode is.

Using the notation in [Knu00], the error-recovery and error-propagation properties of the 2DEM mode could be analyzed using the following equation.

$$P_i = f_k(C_{\left\lfloor \frac{i}{NSB} \right\rfloor * NSB}, C_{\left\lfloor \frac{i}{NSB} \right\rfloor * NSB + BPR}, C_{\left\lfloor \frac{i}{NSB} \right\rfloor * NSB + 2*BPR}, \ldots\ldots$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, C_{\left\lfloor \frac{i}{NSB} \right\rfloor * NSB + (NSB-1)*BPR})$$

Note that i in the equation refers to sub-blocks of size n, and not to 2D blocks of size n^2/8. For $1 \leq i \leq$ M/n, P_i refers to the ith plaintext sub-block, and C_i refers to the ith ciphertext sub-block, while f_k represents some decryption function with a parameter k representing the users' key, and $\lfloor\ \rfloor$ is the floor operator.

According to the equation above and the discussion in [Knu00], 2DEM has finite error-propagation, and the extent of such error-propagation (and accordingly error-recovery) depends on the values of NSB and BPR. NSB is fixed for a certain block cipher of size n, and equals n/8, while BPR is a randomly-generated integer from a set of possible values. The point relevant to error-propagation is that 2DEM uses NSB plaintext sub-blocks that are far apart by BPR sub-blocks to calculate C_i, and similarly uses NSB ciphertext sub-blocks that are far apart by BPR sub-blocks to calculate P_i. This implies that an error in a bit or a sub-block of ciphertext or plaintext propagates to the encryption/decryption of the 2D block containing that bit or sub-block.

The 2D-Encryption Mode, thus, strikes a balance in its error-recovery and error-propagation properties by making the extent of error-propagation finite (for error-recovery properties, which are practically desired) but large (for error-propagation properties, that [Knu00] advocates to be more secure).

V. 2D-Encryption Mode and Some Practical Issues

This section discusses some practical issues related to the 2D-Encryption Mode.

1. The design issue in 2D-Encryption Mode that was affected by practical considerations is choosing the value of NSB. Security, in general, requires a large value for NSB, but on the other hand, smart cards, byte-oriented processors (in embedded systems), and software implementations could not easily deal with bits, as they normally and naturally deal with bytes. Bit manipulation is obligatory if 2DEM is implemented with NSB = n (the largest possible value for NSB), thus making the 2D block of size n^2 bits (instead of $n^2/8$ bits).

Accordingly NSB = $n/8$ was appropriately chosen so that software implementations of 2DEM could do the transposition (compression permutation) between rows and columns, between the first and the second phases of 2DEM, in "no time", using appropriate coding for (byte-addressed) memory accesses; while for hardware implementations of 2DEM these transpositions are typically done in very small time, irrespective of the size of the input and the output of the transposition. Additionally, NSB = $n/8$, keeps the good security behavior of 2DEM, which grows quadratically exponential as n grows. ($O(c^{n^2}$) ciphertext is required to mount a successful birthday attack).

Also, notice that, the maximum value for BPR is M/(n·NSB). That implies that the smaller the value of NSB the larger the range of possible values for BPR. This allows for more flexibility in generating the random BPR.

Furthermore, messages are required to be multiples of n·NSB bits, and so decreasing this value requires less padding for each message. Also the chosen value of NSB lets the 2D-Encryption Mode be more attractive to sector-oriented file-systems, where a disk sector is usually 512 bytes = 2 × 256 bytes = 8 × 64 bytes (if using 2DEM with 128-bit AES, the number of bytes required per block is 256 bytes, while for DES the number required is 64 bytes).

Also, choosing NSB = $n/8$, rather than n, allows a greater potential for parallelization if each n·NSB bits block is encrypted on a different processor, as a smaller value of NSB allows more processors to be used, if available, for example in a distributed or networked system.

2. From a practical point of view, there is some implementational advantage if the encryption and decryption operations of a mode are similar. 2DEM decryption is as simple as 2DEM encryption. It uses the decryption operation of the underlying block cipher, and the row-decryption phase and column-decryption phase are transposed. The column-decryption phase is done first, followed by the row-decryption phase. This could be effected by transposing the 2D data block (a 2D matrix transpose, which is just the same compression permutations used between the row-encryption and column-encryption phases of 2DEM), and performing the row-decryption phase followed by the column-decryption phase (i.e., as in 2DEM encryption, processing of rows is followed by processing of columns), and performing another data transpose. This has the advantage that encryption and decryption could use exactly the same module, especially if the underlying block cipher is the inverse of itself (though, for block ciphers, this may not be recommended from a security point of view, and a slight difference between encryption and decryption is always recommended). DES has this property, and only the order of the subkeys has to be reversed [NBS77]. AES, on the other hand, is not the inverse of itself, but has the decryption and encryption operations *structurally similar* to each other, i.e., both operations use the same components [DR99].

Since the "2D array" transpositions of data, before and after 2DEM is applied, could be done in software and hardware in "no time", this implies that 2DEM decryption could be efficiently implemented with no performance degradation using mainly the same hardware components and software modules of 2DEM encryption, using the fact that DES decryption and AES decryption are similar to DES encryption and AES encryption.

3. An estimate for the time requirements of 2DEM, if implemented on a single processor, is given below. For a message of length M bits, and using an underlying block cipher of size n, the time required for the 2D-Encryption Mode to encrypt the message is:

$$t = 2 \cdot \frac{\mathrm{M}}{n} \cdot c + \tau \cdot \frac{\mathrm{M}}{n \cdot NSB} - \varepsilon \cdot (2 \cdot \frac{\mathrm{M}}{n} - 1)$$

where

c = the time required to do one encryption/decryption using the underlying block cipher,

τ = the time required to do the transpose operation (M/(n·NSB) is the total number of 2D blocks in the message, and NSB = $n/8$ is chosen to let software implementations of the transposition be done in "no time", using appropriate coding for memory accesses; and hardware implementations do transpositions in very small time irrespective of the size of the input and the output), and

ε = the time required to calculate the encryption/decryption subkeys. These need not be calculated more than once for the whole message, so they are subtracted from the time t.

Typically, we have $c >> \tau$ and $c >> \varepsilon$, so $t \cong 2 \cdot \frac{\mathrm{M}}{n} \cdot c$

In accordance with what is mentioned earlier in point 2, we do not distinguish between time requirements of 2DEM encryption and 2DEM decryption; and if the encryption and decryption operations of the underlying block cipher have different time requirements, the constants c, ε could be split to c_e, ε_e and c_d, ε_d.

(In the longer version of this paper more practical issues related to 2DEM are presented).

VI. Concluding Remarks

In this paper, we presented a new encryption mode, the 2D-Encryption Mode (abbreviated 2DEM). We evaluated it from the viewpoints of security, performance, and practicality. We discussed how it is a general-purpose mode, and how it also has special affinity to some types of applications.

References

[ANS83] ANSI X3.106, *American National Standard for Information Systems - Data Encryption Algorithm - Modes of Operation*, American National Standards Institute, 1983.

[BDJR97] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, *A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation*, Proceedings of the 38th Symposium on Foundations of Computer Science, IEEE, 1997.

[BI99] M. Bellare, R. Impagliazzo, *A tool for obtaining tighter security analysis of pseudorandom function based constructions, with applications to PRP → PRF conversion,* Manuscript, 1999.

[BKR94] M. Bellare, J. Killian, P. Rogaway, *The Security of Cipher Block Chaining,* Advances in Cryptology - Crypto 94 Proceedings, 1994.

[BKR98] M. Bellare, T. Krovetz, P. Rogaway, *Luby-Rackoff Backwards: Increasing Security by Making Block Ciphers Non-Invertible*, Advances in Cryptology--Eurocrypt'98 Proceedings, Springer-Verlag, 1998.

[DR99] J. Daemen, V. Rijmen, *AES Proposal: Rijndael,* AES *Algorithm* Submission, 1999.

[HWKS98] C. Hall, D. Wagner, J. Kelsey, B. Schneier, *Building PRFs from PRPs,* Advances in Cryptology -- Crypto 98 Proceedings, Springer-Verlag, 1998.

[IASDM97] *Intel Arch. Software Dev. Manual: Basic Architecture, Order Number 243190*, 1997.

[ISO97] ISO/IEC 10116, *Modes of Operation for an n-bit Block Cipher Algorithm*, 1997.

[Jut00]C. S. Jutla, *Encryption Modes with Almost Free Message Integrity,* Preprint, 2000.

[Knu00] L. R. Knudsen, *Block chaining modes of operation,* Reports in Informatics, Report No 207, ISSN 0333-3596, University of Bergen, Norway, 2000.

[LRW00] H. Lipmaa, P. Rogaway, D. Wagner, *CTR-Mode Encryption,* Comments to NIST concerning AES Modes of Operation, 2000.

[NBS77] *National* Bureau of Standards, NBS FIPS PUB 46, *Data Encryption Standard,* US Department of Commerce, 1977.

[NBS80] National Bureau of Standards, NBS FIPS PUB 81, *DES Modes of Operation*, US Department of Commerce, 1980.

[NR97] M. Naor, O. Reingold, *On the Construction of Pseudo-Random Permutations: Luby-Rackoff Revisited, Proceedings* of the 29th Annual Symposium on Theory of Computing, ACM, 1997.

[Rog00a] P. Rogaway, *OCB Mode: Parallelizable Authenticated Encryption,* Comments to NIST concerning AES Modes of Operation, 2000.

[Rog00b] P. Rogaway, *PMAC: A Parallelizable Message Authentication Code,* Comments to NIST concerning AES Modes of Operation, 2000.

[RRY00] R. L. Rivest, M. J. B. Robshaw, Y. L. Yin, *The case for RC6 as the AES,* AES Round 2 public comment, 2000.

[Sch95] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Wiley, 2nd Edition, 1995.

Appendix

Sample Images

Given below are two sample images, together with images resulting from their encryption. Following each image (or encrypted image) is the number of colors used in the image, and the image's histogram.

The "1D" images represent the ECB DES encryption of the Original Image, using the key `1234567890ABCDEF`, while the "2D" images represent the 2D-Encryption Mode encryption of the Original Image, using DES with the key `1234567890ABCDEF`. (BPR was not chosen randomly, but *for demonstration purposes* equals number of pixels in a row divided by NSB = `numcols`/8)The histogram of each image (or encrypted image) shows the following:

i. A graph showing the number of occurrences (on the ordinate) of each of the colors (0..255 on the abscissa) in the image. The variance between the ordinate values could be seen visually from the graph.

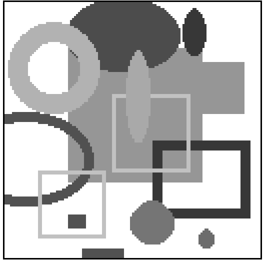
ii. The maximum number of times a single color is repeated in the image. For example, for First Sample's Original Image Histogram this is 7607. Next to it, is shown (in parentheses) the percentage of this number to the total number of pixels of the image.

iii. The minimum number of times a single color is repeated in the image. For example, for First Image's Original Image Histogram this is 0, and it means that some color does not exist in First Sample's Original Image.

iv. The average color of the image. This is calculated by multiplying each color value by the ratio in which it is repeated in the whole image. For example, for First Sample's Original Image Histogram this is 187, and that means that the average color of First Image's Original Image is equivalent to a gray-scale color of 187, i.e., a somewhat light image (nearer to the "white" end of the range of colors).

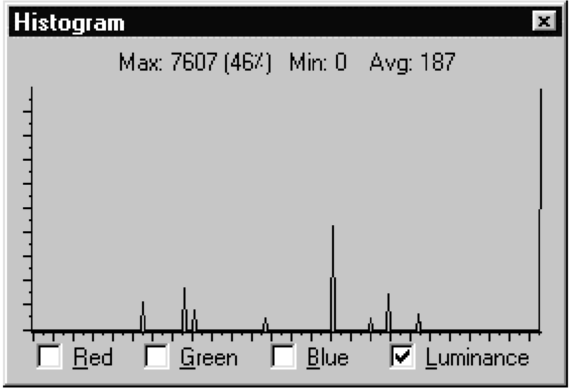
First Sample

Original Image



Colors Used = 10

Original Image Histogram

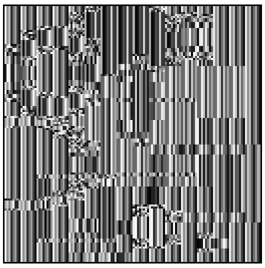


Histogram

Max: 7607 (46%) Min: 0 Avg: 187

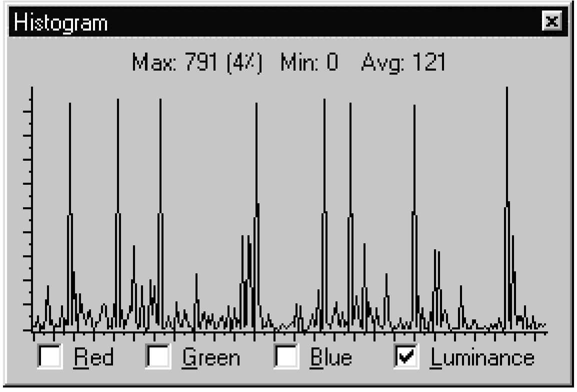
☐ Red ☐ Green ☐ Blue ☑ Luminance

1D



Colors Used = 254

1D Histogram

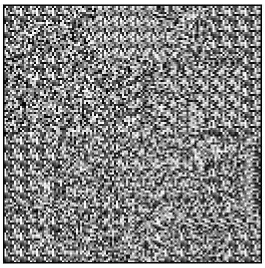


Histogram

Max: 791 (4%) Min: 0 Avg: 121

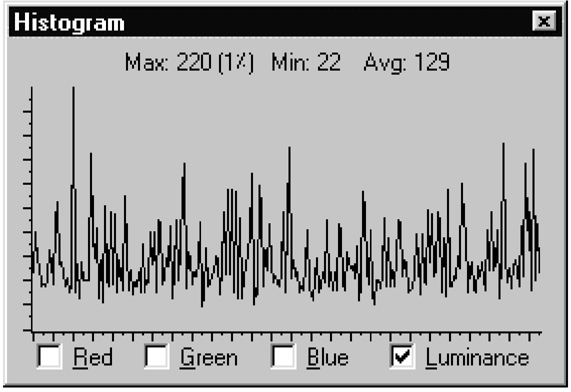
☐ Red ☐ Green ☐ Blue ☑ Luminance

2D



Colors Used = 256

2D Histogram



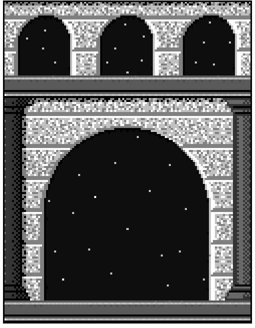
Histogram

Max: 220 (1%) Min: 22 Avg: 129

☐ Red ☐ Green ☐ Blue ☑ Luminance

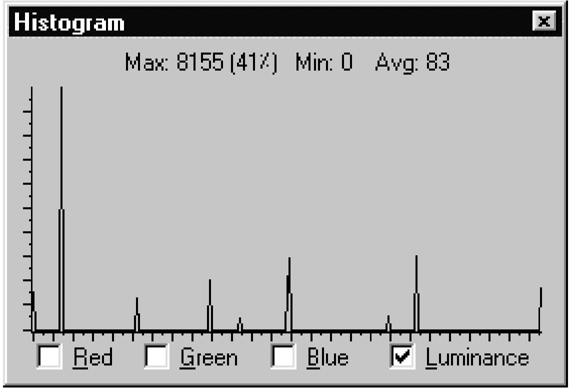
Second Sample

Original Image



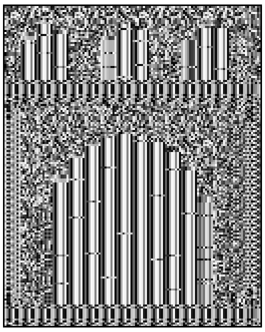
Colors Used = 9

Original Image Histogram



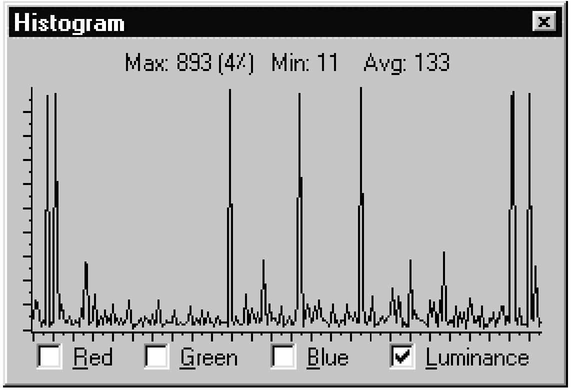
Max: 8155 (41%) Min: 0 Avg: 83

1D



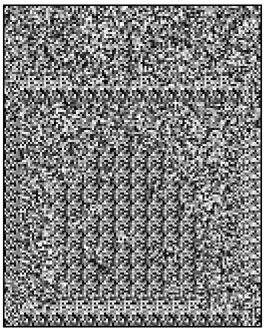
Colors Used = 256

1D Histogram



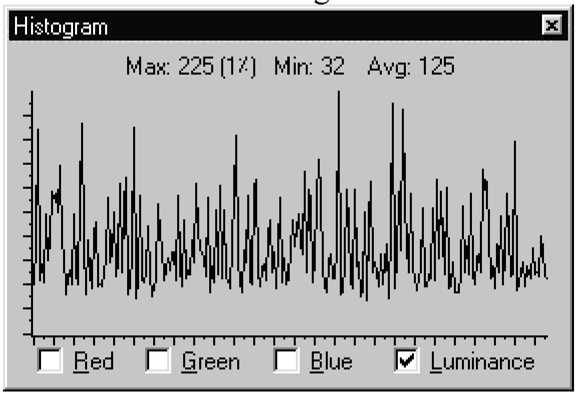
Max: 893 (4%) Min: 11 Avg: 133

2D



Colors Used = 256

2D Histogram



Max: 225 (1%) Min: 32 Avg: 125