OLMEC Encryption Algorithm Designed and Implemented by James Pate Williams, Jr. © 1995 - 2010 All Rights Reserved

Algorithm Parameters

NUMBER_OF_ROUNDS = {1, 2, 3,...}
NUMBER_OF_TABLES = 8
BITS_PER_INT = 32
BITS_PER_LONG = 64
BYTES_PER_SEGMENT = 64
LONGS_PER_SEGMENT = 8
FILE_LENGTH
KEY_0
KEY_1

Algorithm Description

The algorithm acts on a block of 64-bits that is viewed either as an unsigned long integer or a series of bytes. Another grouping is called a segment and is viewed either as unsigned long integers or bytes. There are LONGS_PER_SEGMENT blocks in a segment.

Table and Key Generation

Step 1G - Generate NUMBER_OF_TABLES bit permutation and segment scrambling tables. The bit permutation tables are used to permute the bits in a given block, e.g. an unsigned long integer. The scrambling tables act on segments and are also actually permutation tables. The bit permutation tables are of length BITS_PER_LONG and the scramble tables have BYTES_PER_SEGMENT entries. . Generate randomly or specify two 64-bit unsigned long keys.

Key Schedule

There is one set of encryption keys per round which are generated by rotating KEY_0 one bit left per round and KEY_1 one bit right per round and are numbered 0, 1, ..., NUMBER_OF_ROUNDS - 1. The decryption keys are the encryption keys in reverse lexicographic order.

Encryption

Step 1E - Divide the file into segments of FILE_LENGTH / BYTES_PER_SEGMENT. Call the quotient n_s. Calculate the modulus $r_f =$ FILE_LENGTH mod BYTES_PER_SEGMENT. If $r_f \neq 0$ then add an extra segment with BYTES_PER_SEGMENT - r_f random padding bytes. Set $n_s = n_s + 1$.

Step 2E - For each of LONGS_PER_SEGMENT do XOR an unsigned long integer in a segment with key schedule based on KEY_0. Permute each bit in every unsigned long integer in a segment using one of the NUMBER_OF_TABLES bit permutation tables. XOR the resulting unsigned long integer with key schedule based on KEY_1. Change bit permutation table every unsigned long integer.

Step 3E - Scramble (permute) each byte in a segment using a different scramble table for each segment.

Step 4E - Repeat Steps 2E and 3E for NUMBER_OF_ROUNDS -1.

Step 5E - Repeat Steps 2E to 4E until all segments have been encrypted.

Step 6E - Write the bytes "OLME" and the FILE_LENGTH to the encrypted file and finally write all the segments to the file.

Decryption

Step 1D - Read the header consisting of the bytes "OLME" and the FILE_LENGTH.

Step 2D - Unscramble (apply inverse permutation) the BYTES_PER_SEGMENT bytes in a segment using a different scramble table for each segment.

Step3D - For each of LONGS_PER_SEGMENT do XOR an unsigned long integer in a segment with key schedule based on KEY_1. Un-permute each bit in every unsigned long integer in a segment using one of the NUMBER_OF_TABLES bit permutation tables. XOR the resulting unsigned long integer with key schedule based on KEY_0. Change bit permutation table every unsigned long integer.

Step 4D - Repeat Steps 2D and 3D for NUMBER_OF_ROUNDS -1.

Step 5D - Repeat Steps 2D to 4D until all segments have been decrypted.

Step 6D - Write FILE_LENGTH decrypted bytes to the decrypted file.

Padding of a possible final segment was done by choosing random bytes.

Test Files

The first test file, Test384.txt, that we used the file consisting of 384 ASCII characters as illustrated below with 8 '\r' and '\n' combinations hidden. Therefore, we utilized 18 distinct characters. Table 1 shows the plain text character frequencies.

```
0123456789abcdef0123456789abcdef0123456789abcd
0123456789abcdef0123456789abcdef0123456789abcd
0123456789abcdef0123456789abcdef0123456789abcd
0123456789abcdef0123456789abcdef0123456789abcd
0123456789abcdef0123456789abcdef0123456789abcd
0123456789abcdef0123456789abcdef0123456789abcd
0123456789abcdef0123456789abcdef0123456789abcd
0123456789abcdef0123456789abcdef0123456789abcd
```

Value	Frequency	Value	Frequency
10	0.0208	55	0.0625
13	0.0208	56	0.0625
48	0.0625	57	0.0625
49	0.0625	97	0.0625
50	0.0625	98	0.0625
51	0.0625	99	0.0625
52	0.0625	100	0.0625
53	0.0625	101	0.0417
54	0.0625	102	0.0417

Table 1 - Plain Text File Byte Frequencies

3AES-EDE was used for comparison purposes. The OLMEC keys are given in Table 2 and the 3AES-EDE keys are to found in Table 3. The NUMBER_OF_ROUNDS used by the OLMEC code is 8.

Key	Decimal Value
KEY_0	2293562864728595408
KEY_1	4307413245297421442

Table 2 - OLMEC Keys

Key	Hexadecimal Value
Key E	3f1c77c4a76e5af019a3073f51fcada6
Key D	489cb3b2f2172961cb2bca4ed1e28db6
Key E	b203fccdda0f838545fdb0626eacac78

Table 3 - 3AES-EDE Keys

The runtimes for the first test file, Test384.txt are given in Table 4. The runtimes are given in HRS:MIN:SEC:MILLISECONDS format.

Algorithm	Encrypt	Decrypt
OLMEC	00:00:00.003	00:00:00.003
3AES-EDE	00:00:00.051	00:00:00.018

Table 4 - Runtimes for the File Test384.txt

Two other PDF files were used for testing purposes. One was a file of length 125,362 bytes named "Guitar Chords-Scales Interface.pdf" and a much larger file consisting of 28,842,031bytes entitled the "Family Jewels.pdf" whose URL maybe found as shown below:

http://www.foia.cia.gov/search.asp?pageNumber=1&freqReqRecord=FamilyJewels.txt

The runtimes for these two files are illustrated in Table 5 and Table 6 using 3AES-EDE and OLMEC, respectively.

Filename	Encrypt	Bytes/Second	Decrypt	Bytes/Second
Guitar Chords-Scales Interface.pdf	00:00:01.058	118,490	00:00:02.075	60,415
Family Jewels.pdf	00:04:00.194	120,078	00:08:01.111	59,948

Table 5 - 3AES-EDE Runtimes

Filename	Encrypt	Bytes/Second	Decrypt	Bytes/Second
Guitar Chords-Scales Interface.pdf	00:00:00.409	306,508	00:00:00.410	305,760
Family Jewels.pdf	00:01:31.332	315,793	00:01:33.762	307,608

Table 6 - OLMEC Runtimes

Frequency histograms or spectra are shown in Figures 1, 2, and 3, for the unencrypted file, 3AES-EDE encrypted file, and OLMEC encrypted file, respectively, for the "Family Jewels.pdf".

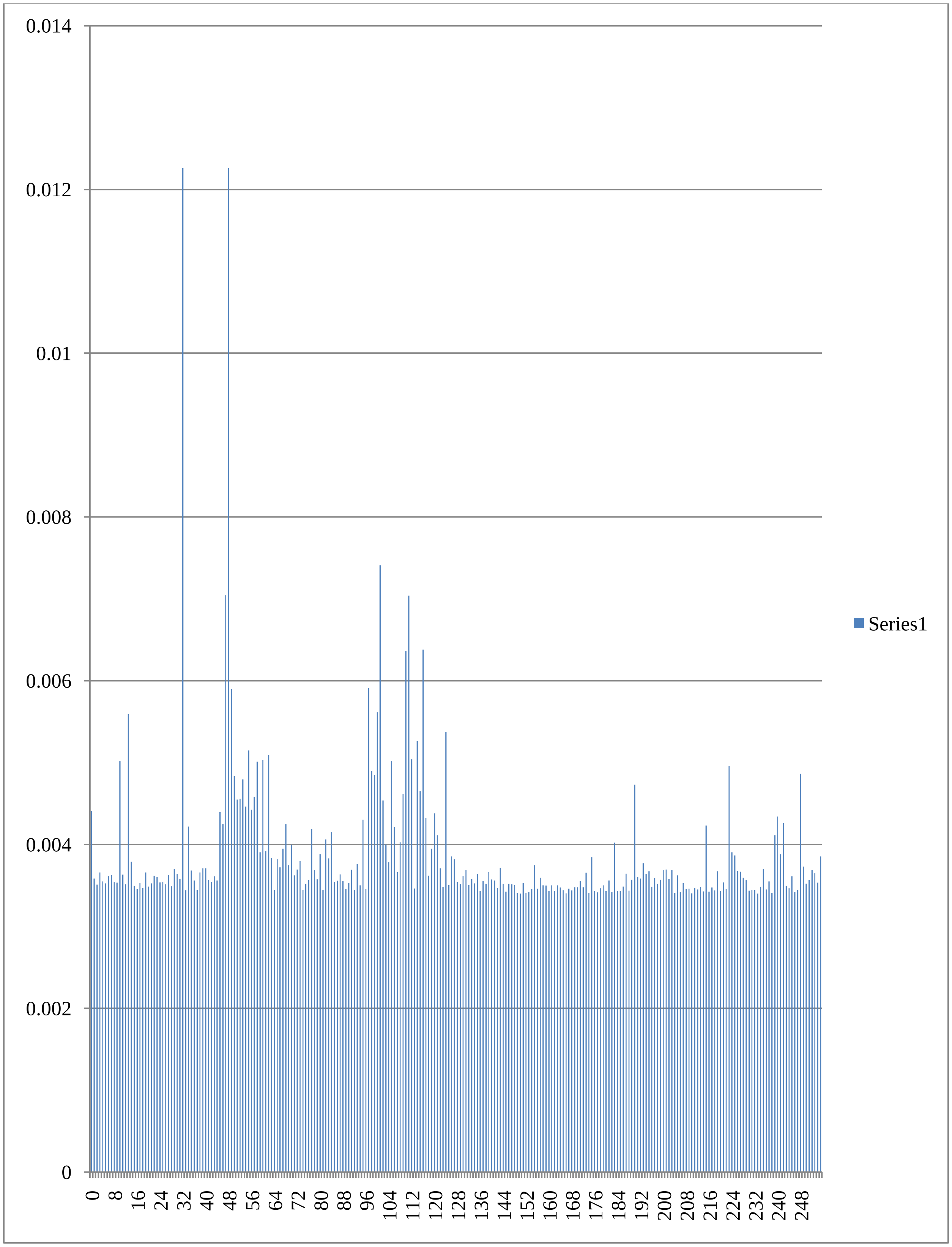


Figure 1 - Family Jewels Unencrypted (Plaintext) Frequency Histogram

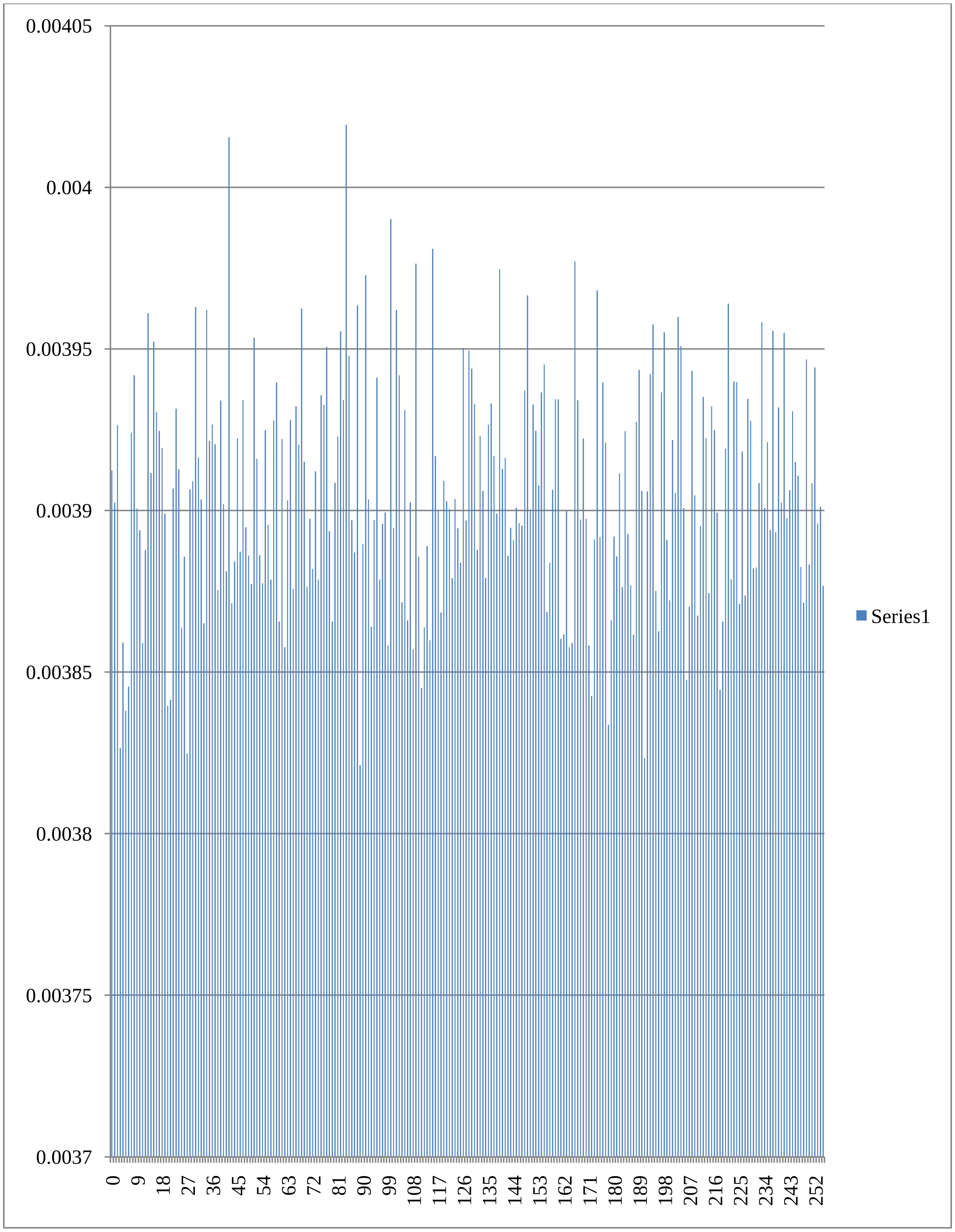


Figure 2 - 3AES-EDE Family Jewels Encrypted (Cipher Text) Frequency Histogram

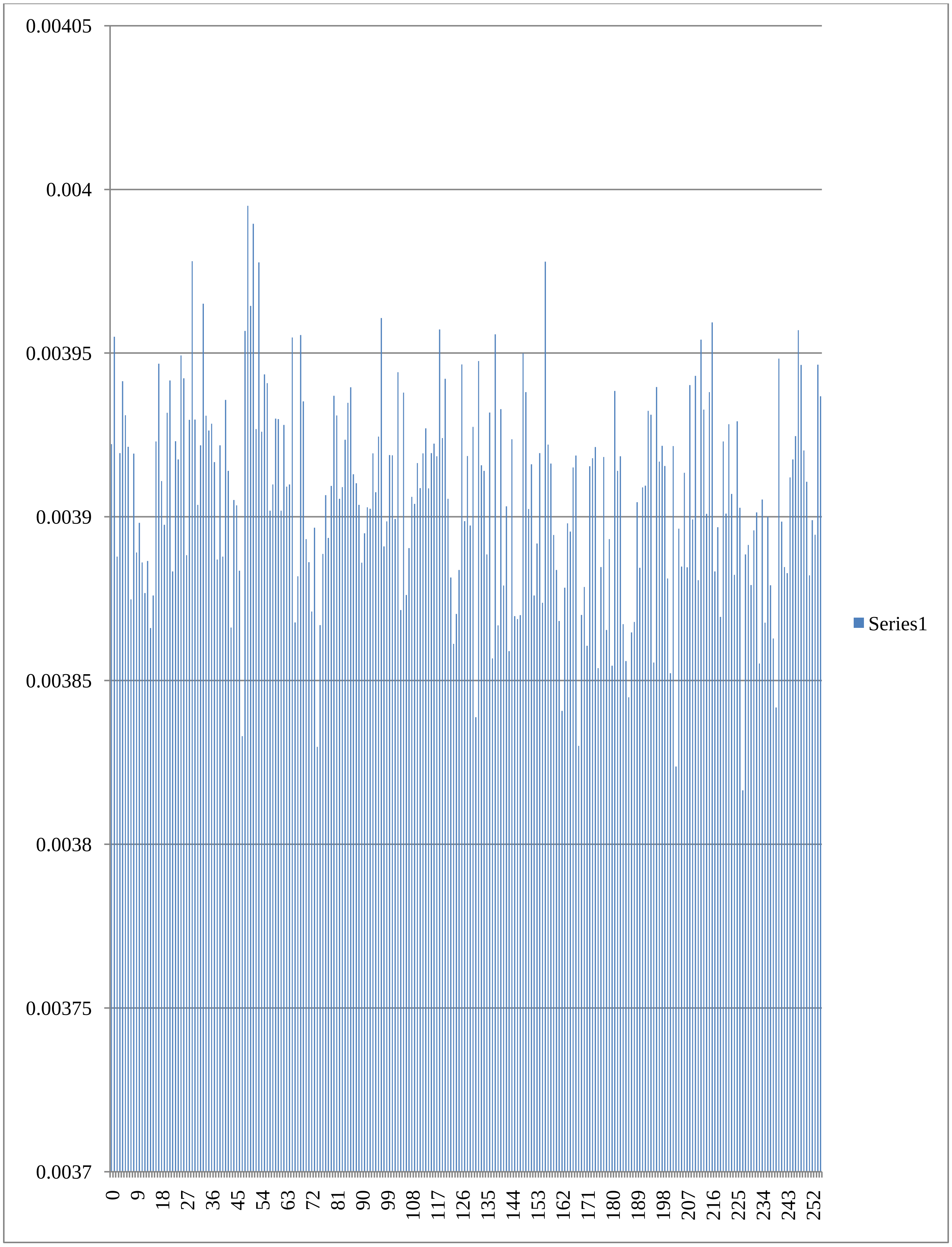
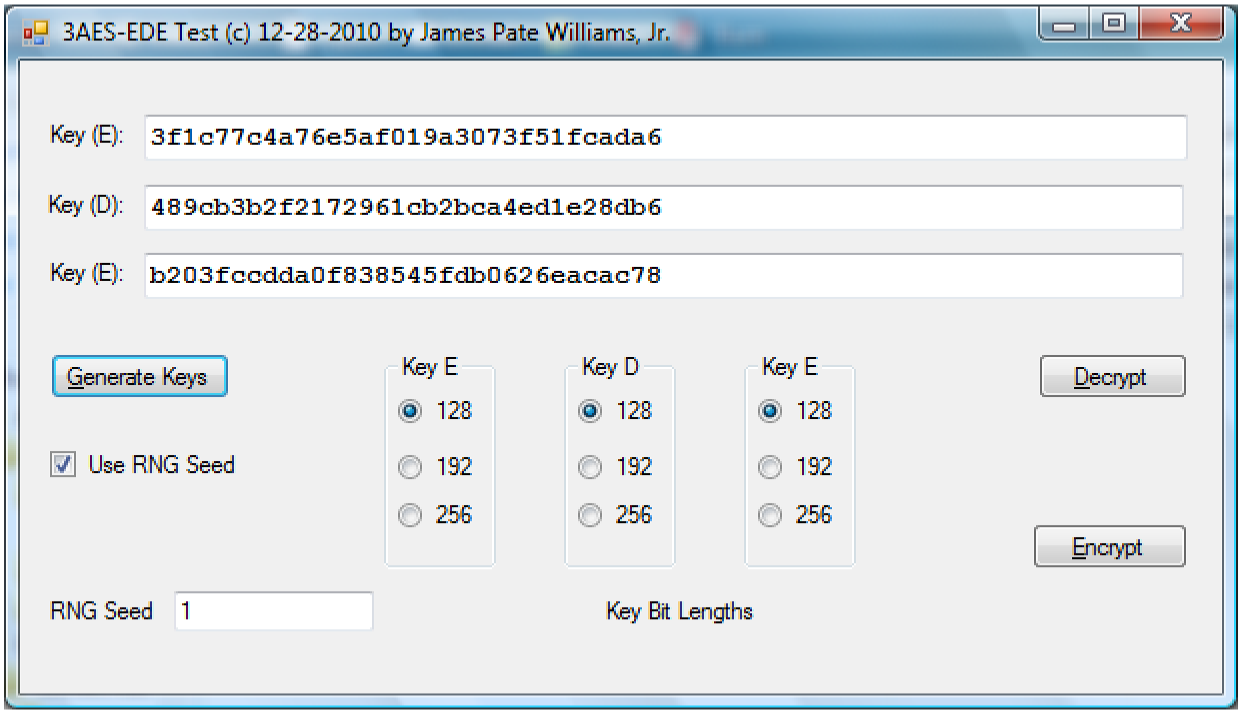


Figure 3 - OLMEC Family Jewels Encrypted (Cipher Text) Frequency Histogram

The two forms used by the 3AES-EDE test program are shown in Figure 4 and Figure 5. The key generation algorithms are unsecure due to usage of a non-cryptographically secure pseudorandom number generator.



3AES-EDE Test (c) 12-28-2010 by James Pate Williams, Jr.

Key (E): 3f1c77c4a76e5af019a3073f51fcada6

Key (D): 489cb3b2f2172961cb2bca4ed1e28db6

Key (E): b203fccdda0f838545fdb0626eacac78

Key E: [✓ 128] 192 256

Key D: [✓ 128] 192 256

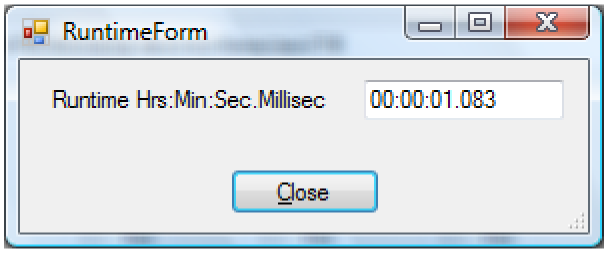
Key E: [✓ 128] 192 256

Key Bit Lengths

[✓] Use RNG Seed

RNG Seed: 1

Figure 4 - 3AES-EDE Test Program Main Form



RuntimeForm

Runtime Hrs:Min:Sec.Millisec: 00:00:01.083

Figure 5 - 3AES-EDE Test Program Runtime Dialog

The OLMEC reference test program utilizes the four forms shown in Figures 6, 7, 8, and 9. The textboxes that appear blue are read only instances.

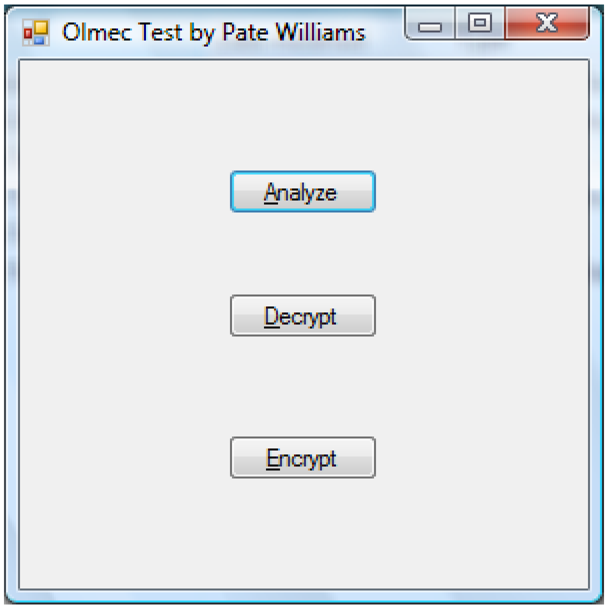
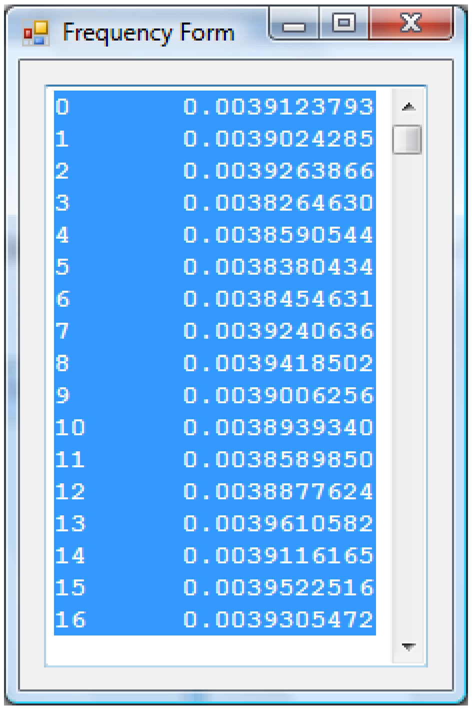


Figure 6 - OLMEC Test Program Main Form



```
0          0.0039123793
1          0.0039024285
2          0.0039263866
3          0.0038264630
4          0.0038590544
5          0.0038380434
6          0.0038454631
7          0.0039240636
8          0.0039418502
9          0.0039006256
10         0.0038939340
11         0.0038589850
12         0.0038877624
13         0.0039610582
14         0.0039116165
15         0.0039522516
16         0.0039305472
```

Figure 7 - OLMEC Analyze Function Frequency Distribution Output

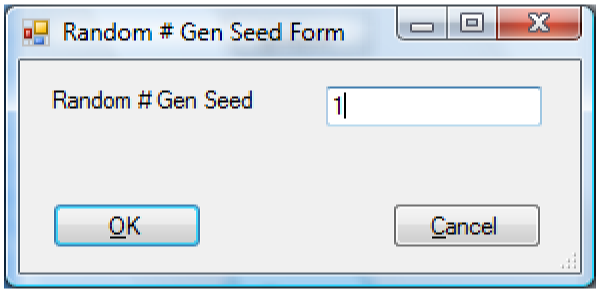
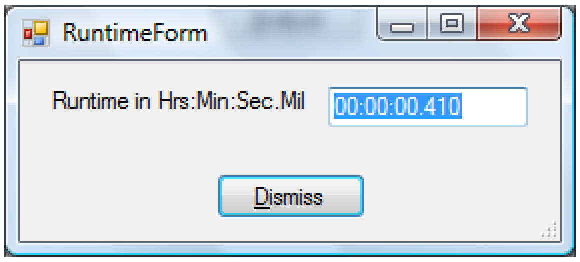


Figure 8 - Random Number Generator Form



Runtime in Hrs:Min:Sec.Mil 00:00:00.410

Figure 9 - OLMEC Runtime Form

The average deviations and standard deviations for the unencrypted file the "Family Jewels.pdf" and the two encrypted files generated by 3AES-EDE and OLMEC are given in Table 7.

File	Avg.	Avg. Dev.	Std. Dev.
Unencrypted	0.003906	0.000529	0.000987
3AES-EDE Encrypted	0.003906	2.79461E-05	3.4933E-05
OLMEC Encrypted	0.003906	2.48948E-05	3.12002E-05

Table 7 - Simple File Statistics for the "Family Jewels.pdf"

If the encrypted file was randomly distributed over 256 distinct byte values then each value in the frequency distribution would be 0.00390625.

The two-tailed dependent T-Test for a statistical difference between the OLMEC and 3AES-EDE frequency distributions for the file the "Family Jewels.pdf" yields a probability of 0.999998919 so there is no statistical difference between the byte frequency mappings. Similar tests for a statistically measurable difference between the OLMEC frequency spectrum and the random distribution for the same file yield a probability of 0.999999866 and same test between 3AES-EDE and the random frequency array yields a probability of 0.99999872. So we can't find any significant statistical differences between the three frequency spectra.